Fra SGML til begrunnede påstander om verden. Et system for analyse av geografiske resonnementer uttrykt i historiske tekster

Hovedoppgave Cand. Philol. i Språk, logikk og informasjon ved Institutt for lingvistiske fag, Universitetet i Oslo

av Øyvind Eide

26. oktober 2004

Innhold

Forord

1

Så lenge historiske tekster har blitt skrevet, har man henvist til andre tekster. Det er vanskelig å tenke seg en tekst uten en eller annen form for intertekstualitet. I det som utviklet seg til å bli vitenskaplige historiske tekster, har denne intertekstualiteten etterhvert blitt formalisert i et kildehenvisningssystem (Grafton 1997).

Nå er en storstilt overgang fra papirbaserte til digitale kilder, verktøy og publisering i gang i humaniora. Hvordan kan nye systemer for kildehenvisninger utvikles i en digital verden samtidig som det grunnleggende vitenskapelige innholdet beholdes?

I denne oppgaven vil jeg undersøke nye måter å løse kildehenvisningsproblemene på i en digital tid. Jeg foreslår et system som kobler sammen kildetekst og analyser, men med et klart skille mellom dem. Jeg har utviklet en prøveimplementasjon som er brukt i analysene i oppgaven.

I tillegg til selve analysene gir programsystemet som beskrives i oppgaven to viktige ting. Det gir så langt som mulig en direkte kobling mellom analysene som gjøres og analyseresultatene på den ene siden, og den digitale versjonen av teksten analysene er basert på på den andre. Det gir også en beskrivelse av alle trinn som er gjort fra systemet leste inn teksten i SGML-format, fram mot analysenes sluttføring. Denne beskrivelsen inneholder de mekaniske bearbeidelsene programmet gjorde, sammen med de beslutningene brukeren tok under prosessen og begrunnelser for disse.

Det betyr at selve teksten, resultatet av analysene, og de valg som ledet fram til analysene kan publiseres som en helhet, samtidig som denne helheten kan deles opp slik at man tydelig viser hvem som er ansvarlig for hvilke deler.

2

1700-tallet, som teksten som brukes som kilde for analysene i denne oppgaven omhandler, var en tid da flertallet av befolkningen i Danmark-Norge levde uten politisk ansvar eller organisasjonsfrihet. Kongen hadde ansvaret for alle politiske valg i eneveldet.

Samer, kvener og nordmenn av lavere klasser i områder under dansk kontroll hadde likevel sine egne agendaer, både som individer og i grupper. De søkte å nå sine mål, ofte på tvers av statens ønsker.

En statsmakt vil alltid motta en strøm av informasjon fra folket. Denne informasjonsstrømmen vokste i kvantitet i Vest-Europa på 1700-tallet (Burke 2000, kap. 6) Ofte går kunnskap til statsmakten i form av forespørsler og bønneskrifter initiert av folk (Sogner 1996, s. 235), men det skjedde også aktiv innsamling av informasjon fra folket initiert ovenfra. I forbindelse med grenseforhandlingene med Sverige fram mot Strømstadtraktaten av 1751 (Strømstadtraktaten 1967), ble det satt igang en storstilt innsamling av kunnskap om grenseforholdene mot Sverige og (selv om dette var under hånden) mot Russland.

Deler av materialet som kom ut av denne undersøkelsen danner kildegrunnlag for analysene i kapittel 3 i denne oppgaven (Schnitler 1962). Det er et materiale der vanlige folk i grensetraktene; samer, nordmenn og kvener, defineres som eksperter. Deres vitnemål blir lyttet til og nøyaktig nedskrevet — *hvor* nøyaktig vil jeg diskutere senere.

1+2

I denne oppgaven undersøkes to ulike ting:

1. Er et integrert system for tekstanalyse slik det beskrives her brukbart?
2. Kan vitnemålene i Schnitlers protokoller brukes til å vise forskjeller mellom hvordan ulike samfunnsgrupper på 1740-tallet tenkte om geografi?

Disse to undersøkelsene er gjort gjensidig avhengige av hverandre. Analysene i 2 bruker verktøy utviklet i 1, mens systemet i 1 bruker analysene i 2 som et case.

I kapittel 1 skisseres et bakteppe for resten av oppgaven, basert på utvalgte emner fra områder som informatikk, filologi, historie og korpuslingvistikk. I kapittel 2 beskrives implementasjonen av programsystemet. I kapittel 3 beskrives analysene i case-studiet, mens kapittel 4 oppsummerer resultatene og gir noen forslag til videre forskning. Tillegg A inneholder programkoden, mens tillegg B inneholder et kjøringseksempel.

Takk til tidligere kolleger ved Universitetsmuseet Tromsø Museum og nåværende kolleger ved Enhet for Digital Dokumentasjon, Universitetet i Oslo, som har gitt innspill og kommentarer til arbeidet helt fra jeg begynte å se på tekster og metoder midt på 1990-tallet. Takk til alle andre jeg har diskutert stoffet med, både innenfor og utenfor akademia. Takk også til veileder Herman Ruge Jervell som tok sjansen på et uklart definert prosjekt, og som med kommentarer og forslag til metoder dyttet det i en fruktbar retning. Takk mest til Heidi, Oda og Jonas som har knyttet meg til andre sider av virkeligheten, noe som ofte viser seg å gi den nødvendige avstand til å angripe stoffet.

Kapittel 1

Bakgrunn

1.1 Referanser og lenker

1.1.1 Kildehenvisningen som historisk fenomen

Kildehenvisninger i historiske framstillinger har blitt brukt fra antikken og framover. Det er snakk om en gammel form hvis bruk ble endret over tid. Sitatene og henvisningene ble etterhvert mer presise som en følge av profesjonalisering. For historiefagets vedkommende skjedde dette mest markant på 1800-tallet (Grafton 1997, s. 29, jfr. kap. 2) Det økte presisjonsnivået hadde også sammenheng med at mens en før-moderne historisk framstilling handlet mer om moralske eksempler, handler en moderne framstilling mer om å få fakta riktig (Burke 2000, s. 182).

Ved bruk av fot-, marg- eller sluttnoter skiller man grafisk mellom to ulike lag i teksten. Dette medfører at det fortelles to parallelle historier (Grafton 1997, s. 23). Den ene er en historisk framstilling med tolkninger og forslag til hva som kan ha skjedd og hvorfor det i tilfelle skjedde, mens den andre utgjør henvisninger til andre tekster som skal underbygge den første historien. En kan også si at den ene historien viser resultatene og den andre forskerens vei fram til resultatene (ibid, s. 200).

En kildehenvisning i en historisk tekst står således der for å vise leseren til de kildedokumenter man baserer seg på, slik at hun kan etterprøve forfatterens påstander. Et eksempel på dette er at når jeg et annet sted i denne oppgaven (i avsnitt 1.6.1) hevder at mange samer kunne lese på 1740-tallet, viser jeg til noen kilder for denne påstanden. Disse kildene er ikke primærkilder, men troverdige sekundære kilder. I noen av dem er det dessuten henvisninger videre til de mest primære kildene som er tilgjengelige. En interessert leser kan da følge henvisningene tilbake og se hva grunnlaget for påstanden er.

Men kildehenvisninger som system virker også i seg selv som et kvalitetsstempel. Ved å henvise gir man teksten en tyngde ved å si "jeg underbygger". Dette gjør også lesere som ikke sjekker kildene — endog lesere som ikke kjenner til kildene — mer overbeviste, på samme måte som kunder overbevises av

håndverkerens svennebrev som henger på veggen, selv om de ikke vet hva en svenneprøve innebærer eller hvordan et ekte svennebrev ser ut (ibid, s. 7).

Kildehenvisningen som metode rommer imidlertid et paradoks som Anthony Grafton beskriver slik, basert på diskusjoner mellom Etienne Pasquier og hans kritikere:

> [A] genuine paradox in the modern routine of documentation, which claims to require that one prove both that each sentence is original and that is has a source. [...] [T]he provision of documentation was more likely to provoke dissent than assent from a modern reader. Cited documents necessarily suggested that a problem could be solved in ways other than that chosen by the historian. (ibid, s. 143)

Det er umulig for en forfatter av historiske tekster å henvise til alle kilder man bruker og underbygge alle fakta i teksten (ibid, s. 233). Man må alltid sette en grense for hva som forutsettes kjent. Man oppgir ikke kilder til når Norge gikk sin grunnlov i en tekst beregnet på norske lesere, men en sørafrikansk verdenshistorie ville kanskje gjøre det. Det er også en grense for hvor marginale kilder man kan ta med — mye av den bakgrunnskunnskapen man øser av er det verken mulig eller ønskelig å henvise til, f.eks. lærebøker man leste for å bli kjent med et saksområde.

1.2 Ekspertsystemer

1.2.1 Klassiske problemer

Ekspertsystemer var et stort satsingsområde innen kunstig intelligens-forskningen på 1970- og 80-tallet (Norvig 1992, kap. 16), (Buchanan 1989). Slike systemer hadde også en viss kommersiell suksess, selv om det etterhvert har vist seg at metoden hadde vesentlige iboende svakheter.

En av hovedgrunnene til at ekspertsystemer ikke fungerte som forventet, var et for enkelt syn på forholdet mellom system og virkelighet. Dette er forsøkt avhjulpet i forskningsretninger som "Embodied cognitive science", som utviklet seg utover 1990-tallet (Pfeifer 2000, s. xii).

I denne oppgaven er slike problemer mindre, fordi det utsnitt av virkeligheten vi forholder oss til er et tekstunivers, noe som medfører at fundamentale problemer med kunstig intelligens-systemer ikke oppstår.[1] En kan si at det man ikke klarer å implementere i slike systemer ikke behøves i vårt eksempel fordi den eneste verden vi har er en tekstlig. Studieobjektet er i utgangspunktet en forenklet modell av verden. "Ekspertene" ekspertsystemet baserer seg på blir således stemmer i en tekst. Våre "eksperter" er tekstfragmenter.

[1] En gjennomgang av slike problemer finnes hos Rolf Pfeifer et.al. (Pfeifer 2000, ss. 59–78). Problemene oppstår når en modell av verden i et dataprogram skal forholde seg til den omskiftelige fysiske virkeligheten.

1.2.2 Hva jeg har hentet fra ekspertsystemene

Systemet som er laget her er ikke et ekspertsystem, men det baserer seg i visse henseende på metoder fra ekspertsystemene, og er delvis inspirert av ekspertsystemer. Et ekspertsystem kjenntegnes ved følgende punkter:

- Områdespesifikke metoder og kunnskap
- Trekker slutninger basert på usikkerhet
- Fleksibel flytkontroll
- Forklarer slutningene

Det er neppe fornuftig å forsøke å bruke tradisjonell ekspertsystemmetodologi som sådan til analyser av historiske tekster: "To the extent that a problem i open-ended (or "open-texured," i.e. requires reasoning about unbounded lists, such as the intended meanings of a sentence), it is *not* a good candidate for an expert system." (Buchanan 1989, s. 157).

Men det siste punktet ovenfor, å forklare slutninger, er viktig også for oss. Buchanan m.fl. skriver i sin oppsummering av fellestrekk ved fungerende ekspertsystemer: "An expert system is a computer program that: [...] *Explains* or makes understandable both what it knows and the reasons for its answer." (ibid, s. 151) I følge Norvig gir dette større tillit hos brukeren enn andre systemer som Prolog: "A system that can explain its solutions to the user in understandable terms will be trusted more." (Norvig 1992, s. 531)

I analyser av tekster er koblingen til selve teksten fundamental for tilliten til en forsker. En forsker i kildebaserte fag som historie, arkeologi eller litteraturfag som ikke kan dokumentere dekning for sine påstander i kildene, har tapt i utgangspunktet — et aktuelt eksempel på dette er Thor Heyerdahl, hvis bok *Jakten på Odin* ble beskyldt for å feiltolke og forvrenge kildene. Når slike påstander kan sannsynliggjøres i den grad at de fester seg, vil resultatene man har publisert kunne avfeies som spekulasjoner og eventyr.

Ingen historiker er avhengig av ekspertsystemer for å oppgi skikkelige kildehenvisninger. Det finnes en lang kulturell tradisjon for å bruke fotnoter og lignende metoder til å henvise til kilder (Grafton 1997). Men formen på kildene og formen på vitenskapelige arbeider er i endring. Vi må utvikle nye metoder for å ta med oss ideene bak kildehenvisningen inn i en digital tid. Vi må sørge for at ideene bevares, og, dersom det er mulig, gjøre tekster etterprøvbare i større grad enn før. En måte å gjøre dette på er å ta med seg ideene om mekaniserte forklaringer fra ekspertsystemene inn i arbeidet med å analysere kildetekster.

I denne oppgaven vil jeg gjøre analyser som forsøker å vise hvordan ulike samfunnsgrupper resonnerte om geografiske forhold midt på 1700-tallet, basert på samtidige kilder, ved hjelp av datamaskinell analysestøtte. Systemet for analysestøtte baserer seg på tradisjonene fra ekspertsystemer. Dette innebærer at systemet arbeider ut fra et sett utledningsregler, og at brukeren spørres når systemet ikke har noen utvetydige regler å følge. Behandling av usikkerhet i slutninger, som var en viktig del av ekspertsystemene, er ikke brukt her.

1.3 Tekstanalyse

I flere fag arbeider man med digitale tekster for å finne ut mer om hvordan språk brukes. Korpuslingvistikk som fag baserer seg på at et ords betydning vises gjennom dets omgivelser (Firth 1957, s. 11). Man bruker også digitale tekster til å studere stil i litterære tekster, ikke minst er dette viktig når man søker å finne ut hvem som er forfatter til bestemte tekster (Oakes 1998, kap. 5). Det er også gjort noe i retning av stilstudier av tekster som ligner på kildematerialet i denne oppgaven. Eksempler på dette finnes i avsnitt 1.6.3.

I historisk forskning er derimot tekstanalyse lite brukt (Thorvaldsen 1999, ss. 37–38 og 91), selv om f.eks. forfatterattribusjon kan ha betydning for historikere gjennom sin påvirkning av kildeforståelse, og i visse tilfelle brukes som metode i historieforskningen. Et eksempel på dette er David Holmes analyse av mormonske skrifter (Holmes 1991). Andre, f.eks. Mark Olsen, omtaler direkte bruk av tekstanalyse for historikere, men konsentrerer seg om å analysere tema og lete etter meningsinnholdet i tekstene, ikke stilanalyse (Olsen 1988). Det ligner således på tradisjonelle (papirbaserte) historiske nærlesninger, som f.eks. Knut Kjeldstadlis analyse av en Marcus Thrane-tale (Kjeldstadli 1999, s. 184 ff.).

1.3.1 Stilistisk analyse

Stilistiske metoder har lenge vært brukt til forfatterattribusjon. Her er det utviklet en rekke statistiske metoder for å skille ulike forfattere fra hverandre for å vise hvilke tekster som har samme forfatter (Oakes 1998). Dersom slike stilforskjeller kan brukes til forfatterattribusjon, betyr det at dersom det finnes stilforskjeller mellom ulike stemmer i teksten, bør man kunne gjenfinne dette i stilforskjeller mellom ulike deler av teksten. En slik undersøkelse er gjort av Larry Stewart for å finne forskjeller mellom ulike stemmer i en og samme litterære tekst, Daniel Defoes *The Fortunes and Misfortunes of the Famous Moll Flanders* (Stewart 2004).

1.3.2 KWIC-konkordanser

Konkordanser har blitt laget siden midten av 1200-tallet (McCarty 1993), mens maskinelt produserte KWIC-konkordanser (Key Word In Context) første gang ble beskrevet i 1959 (Luhn 1966). En KWIC-konkordans lages basert på et søk mot et tekstkorpus, hvor resultatet returneres i ei liste med den eller de ordformene som representerer treff for søket i midten og noen ords kontekst foran og bak. En slik KWIC-konkordans kan sorteres, gjerne på høyre-kontekst, dvs. ordene til høyre for søkets treff (Oakes 1998, s. 151).

Slike konkordanser brukes blant andre av leksikografer. En konkordans viser i hvilke sammenhenger et ord pleier å forekomme. Dette sier noe om ordets betydning. Et ords bruk, og derav også betydning, viser seg i hvilke ord det står i nærheten av (Firth 1957, s. 11). Det er også et oppsett som egner seg til å returnere treff fra søk i litterære tekster.[2]

[2]Flere av de digitale kildene hos Dokumentasjonsprosjektet presenterer søkeresulta-

1.3.3 Datasystemer for import og analyse av tekst

Ideen om å bruke datamaskiner til produksjon av trykte konkordanser oppsto på 1940-tallet (Busa 1998), mens på 1970-tallet kom det første konkordanssystemet for interaktiv bruk, ARRAS (Lancashire 1986, s. 55). Bruk av interaktive konkordansprogrammer førte med seg at sluttbrukere begynte å laste data inn i datamaskinen for å analysere dem der, i stedet for å bruke trykte konkordanser der utgiverne velger et antall måter å sortere og gruppere teksten på, som så trykkes på papir.

Mens 1970 og 1980-tallet var preget av sentrale datamaskiner som ble brukt til batch-generering som gjerne resulterte i trykte konkordanser, f.eks. TUSTEP (TUSTEP WWW), kom det på 1980 og 1990-tallet en rekke PC-baserte verktøy som egnet seg for interaktiv bruk, f.eks. TACT (Bradley 1991). Fra slutten av 1990-tallet kom applikasjoner som kan brukes på verdensveven, slik at man ikke trenger å installere noe eget program på sin datamaskin, men kan bruke en vanlig vevleser. Et eksempel på dette er HyperPo (Sinclair 2003). Et mål for flere forskere og utviklere er nå å gjøre selve analyseapplikasjonene nettbaserte. Man forsøker å modularisere dem og slik gjøre det mulig for en forsker å plukke de analysepakkene hun trenger og de tekstene hun trenger, koble dette sammen og stille sine spørsmål (Bradley 2000), (Sinclar 2004). Jeg vil komme tilbake til dette i avsnitt 4.1.2.

Verktøy for analyse av tekst er også utviklet i en annen tradisjon, innenfor samfunnsvitenskapene. I såkalte CAQDAS-systemer (Computer-Assisted Qualitative Data Analysis Software) legges det gjerne vekt på å knytte innholdsbeskrivende annoteringer, emneord eller koder til tekstsegmenter ("code-and-retrieve"). Å knytte taler til avsnitt, som jeg gjør i kapittel 3, kan minne om dette. Noen av systemene kan også brukes til å bygge opp konseptuelle strukturer basert på kodene (Fielding 2002).

I arbeid med historiske kilder er det en lang tradisjon både for både kvalitativ og kvantitativ analyse. Et tidlig eksempel på et CAQDAS-lignende datasystem for historikere er CLIO/$\kappa\lambda\epsilon\iota\omega$ (Thaller 1985), (Kleio WWW). I mange tilfelle har imidlertid ikke kildemateriale blitt digitalisert i sin helhet, men utvalgte data har blitt skrevet inn i en database basert på kildene. Det betyr at man ikke inkluderer en digital versjon av kildene i analysesystemet, men en ekserpering basert på forskerens vurdering av hva som er det viktige i kildene (Thorvaldsen 1996, s. 15).

Selv om resultatet man kommer fram til kan være det samme, er det en metodologisk forskjell mellom å analysere en tagget tekst og å annotere en tekst. En av forskjellene er at annoteringssystemer gjerne ikke har en hierarkisk ordning av annoteringene. Arbeid er i gang med å lage systemer som baserer seg på kombinasjonen av et XML'sk hierarki og ikke-hierarkiske annoteringer, f.eks. ved å bruke "Layered Markup and Annotation Language" (Czmiel 2004).

Gunnar Thorvaldsen har foreslått et system for å unngå redundante data som baserer seg på Microsoft Windows-baserte metoder for automatisk kobling

ter slik, f.eks. den litterære tekstsamlingen, Qvigstads brev og Renbeitekommissionen (Dokpro WWW).

av datasett i ulike applikasjoner, så som regneark, databaseprogram og statistikkpakker (Thorvaldsen 1999, ss. 196-199). Dette er nokså likt tankene om nettbaserte analyseapplikasjoner som er beskrevet ovenfor, men det er problematisk at løsningen er avhengig av at man bruker operativsystem fra én bestemt leverandør.

Programsystemet jeg lager i denne oppgaven er et interaktivt analysesystem. Hele teksten, slik den finnes i papirbasert forelegg, er med hele veien fram i den forstand at en digital versjon av originalteksten er lagret i informasjonssystemet.

1.4 Digitale utgaver

Det foregår i dag en utstrakt digitalisering av ulike former for tekster, både fag- og skjønnlitterære. Dette inkluderer også historiske kildeskrifter, som kildematerialet i denne oppgaven er et eksempel på. Det ble digitalisert som en del av Dokumentasjonsprosjektet på 1990-tallet.

1.4.1 Utgavetyper

Når man utgir en gammel tekst, gjør man en rekke valg. Dersom man baserer seg på et historisk forelegg, kan man velge å legge inn endringer i de tekstvitnene man bruker, eller man kan gjengi en historisk tekst uendret. En uendret tekst kan så gjengis som en faksimileutgave eller en diplomatisk transkribert utgave (Tanselle 1995, s. 11). I Dokumentasjonsprosjektet var en mye brukt metode å lage diplomatariske tekstutgaver av tidligere trykte bøker, hvor en gjenga det språklige innholdet i en trykt tekst eksakt. Tekstsamlingen som danner grunnlaget for denne oppgaven (Schnitler 1962) ble behandlet slik. Imidlertid var dette tekstlige forelegget selv en utgave av et håndskrift. Denne utgaven var også en diplomatariske utgave laget av ulike utgivere på 1900-tallet.

1.4.2 SGML og XML

SGML er en standard for koding av tekstlig materiale (ISO-8879) som brukes mye til koding av kulturhistorisk materiale, selv om dette ikke er hovedanvendelsesområdet for standarden. Den er i dag best kjent i humaniora gjennom to sett av databeskrivelser (DTD'er), HTML (HTML 1999) og TEI (TEI 2002). Den ekte delmengden XML (XML 2004) er også bedre kjent i dag.

SGML-standarden spesifiserer hvordan man kan lage grammatikker for å beskrive dokumenttyper. Arbeidet som ledet fram til SGML ble påbegynt på slutten av 1960-tallet, mens standarden ble publisert første gang i 1986. Et SGML-dokument består av en databeskrivelse og et dokument. Databeskrivelsen angir et sett av elementer som kan inneholde andre elementer og tekst (`PCDATA`), og beskriver hvordan elementene forholder seg til hverandre. Basert på databeskrivelsen kan man så maskinelt parsere et dokument og finne ut om det syntaktisk er strukturert i henhold til databeskrivelsen. I et normalisert

SGML-dokument markeres alle ikke-tomme elementer med start- og sluttagger. Dokumentet jeg jobber med i denne oppgaven er normalisert.

TEI (TEI 2002) er et sett av databeskrivelser for kulturhistorisk materiale. Disse databeskrivelsene standardiserer hvilke navn de ulike elementene har og hvordan de forholder seg til hverandre. Dokumentet denne oppgaven behandler er ikke kodet i henhold til TEI, men det er kodet i SGML. Vår databeskrivelse er gjengitt i sluttrapporten fra delprosjektet i Dokumentasjonsprosjektet (Eide 1998 A, s. 77–85).

XML (XML 2004) er en forenklet versjon av SGML som ble utarbeidet i første versjon på slutten av 1990-tallet. XML innfører begrepet "velformet" om dokumenter som ikke har noen databeskrivelse det valideres mot, men som følger de grunnleggende reglene for normaliserte XML-dokumenter, slik som at tagger har riktig syntaks og at elementer ikke overlapper hverandre. Mengden av validerte dokumenter er en delmengde av mengden av velformede dokumenter.

1.4.3 Hvorfor digitale utgaver?

Hvorfor lager man digitale utgaver? Dokumentasjonsprosjektets overordnede målsetning var "å gi en samlet og rasjonell tilgang til informasjon om språk og kultur" (Dokpro 1998, s. 6). Målet var altså å gjøre materialet tilgjengelig. Siden det i vårt tilfelle er snakk om ei trykt bok som allerede var tilgjengelig for forskere og andre interesserte, må det være snakk om en *forbedret* tilgang. I utdypingen ligger det en antydning av metodene man ønsket å bruke:

> Tanken var med utgangspunkt i kulturhistoriske og språklige samlinger ved norske universiteter å gjøre kulturrelatert informasjon tilgjengelig for forskere, forvaltning og allmennhet i en form som ville gjøre det mulig å utnytte informasjon fra ulike registre samtidig. Man ville legge vekt på en samlet fremgangsmåte innen de ulike fagfeltene og institusjonene slik at man unngikk å ende opp med et sett av ulike tekniske og faglige løsninger: Det ble også sett som viktig å velge en mest mulig leverandøruavhengig måte å lagre dataene på, og dataløsningene skulle integreres i fagavdelingenes daglige virke. (ibid.)

Her ligger det implisitt at det er snakk om to ulike former for tilgjengeliggjøring: Den ene er å gi faktisk fysisk tilgang. Forskjellen mellom bruk av den analoge originalen og den digitale versjonen er forskjellen mellom forskningsbiblioteket og datamaskinen hjemme eller på kontoret. En liten forskjell for mange forskere, en større forskjell for en reindriftssame eller en lokalhistoriker.

Det andre er å gjennom nye metoder gi bedre tilgang til den informasjonen som finnes i dokumentet, gitt at man har det foran seg. Denne oppgaven dreier seg i hovedsak om dette aspektet. I tillegg ser vi et ønske om krysskobling av informasjon fra ulike samlinger. Dette vil jeg komme inn på i avsnitt 4.2.2.

1.4.4 Lesemåter

En tekst er fysisk tilgjengelig for en leser når symbolene (skrifttegnene) som utgjør teksten er synlige. Teksten er da lesbar. Men for at lesning skal finne sted, kreves det også en språklig forståelse av hva tegnene betyr. Med "leser" mener jeg således en person med kompetanse til å forstå en tekst skrevet på dansk slik det ble brukt midt i det 18. århundre. Leseren forutsettes også å ha en grunnleggende ide om den historiske konteksten teksten ble laget i.

Tekster leses på ulike måter. Et sett av lesemåter som er typiske for lesning av skjønnlitterære tekster kjenntegnes av å følge en historie for historiens egen skyld, av begjær etter estetisk opplevelse, spenning, eller andre følelsesreaksjoner. Ønsket om opplysning og kunnskapstilegnelse er svakere enn i annen lesning, og sees gjerne på som en bieffekt. Dette vil jeg kalle en skjønnlitterær lesemåte.

Lesning av faglitterære tekster er ulik lesemåten beskrevet ovenfor. Man leser snarere på tross av enn på grunn av god historiefortelling, intellektuelt utbytte prioriteres gjerne på bekostning av følelsesmessige reaksjoner. Man leser for å lære. Naturlig nok er dette en lesemåte som Dokumentasjonsprosjektets målsetning legger opp til. Jeg vil kalle det en faglitterær lesemåte.

Historiske kildetekster leses på begge disse måtene. Noen tekster er en del av den skjønnlitterære kanon og faglitterær lesning er en bi-effekt, for andre er det motsatt. Enkelte tekster er innenfor kanon både i skjønnlitterær og faglitterær lesning.

Tekstene som tas opp i denne oppgaven er i liten grad gjenstand for en skjønnlitterær lesning. Dette er ikke på grunn av historien som fortelles — en reise til fots fra Røros til Øst-Finnmark på 1740-tallet kan skildres på en måte som gjør teksten populær som skjønnlitterær tekst. Det er formen som gjør det naturlig å lese den på en faglitterær måte. Selv om alle tekster kan leses på begge måter, er det likevel språklige forskjeller som gjør tekster mer faglitterære eller mer skjønnlitterære. Schnitlers protokoller er således faglitterære tekster.

1.4.5 Tilgjengeliggjøring

Vi har altså foran oss en dokument som inneholder en fagtekst. Den er digitalisert for å gjøre den mer tilgjengelig. Vi har fastslått at teksten gjennom dette er lettere fysisk tilgjengelig. Men i hvilken grad er informasjonen i teksten lettere å få tak i for en leser?

Schnitlers protokoller kjenntegnes ved at de inneholder store mengder konkrete opplysninger. Sannhetsverdien av disse vil diskuteres senere (se avsnitt 1.6.3). De konkrete opplysningene finner man normalt ikke ved å lese hele teksten. Når man er ute etter opplysninger om Tydal, leter man seg fram til seksjonene om Tydal og leser det som står der. Det kan gjøres i manuskriptet, men i den trykte versjonen har man hjelp av et stedsnavnregister som man kan slå opp i. Det viser følgende om Tydal:

> *Tydalen, Tydahls bøjden* 34, 36, 38 ff. 42-48, 50-57, 60, 63, 67 f. 72, 75, 93, 118 f.

Tydalens Annex 36, 39 f. 53, 65, 118.
Tydals Fieldene, di Tydalsche Fielde 47, 50 f. 53, 55, 59 ff. 63, 73, 144.
Tydals kiercke 73. (Schnitler 1962, s. 469)

I den digitale teksten kan man søke i fritekst. Det betyr at man kan søke opp de ulike formene av Tydal og finne de samme henvisningene som ovenfor, bare noe mer finmasket — man slipper å lete på en side, men kommer rett til ordet. I tillegg er Tydal nevnt i innledningen på side XXI, det er ikke med i registeret.

Det er vanskeligere å finne opplysninger i den trykte teksten som ikke er inkludert i et av registrene. Dersom man ønsker å finne alt som handler om ulv er det ikke noe register man kan slå opp i. Eneste løsning er å lese gjennom dokumentet. I et digitalt dokument kan man søke i fritekst, og finner raskt ut at ulver omtales på 13 ulike sider. Dette utgjør en betydelig innsparing for en forsker som ønsker å finne hvor ulver omtales i et sett av dokumenter, og kan godt avgjøre om undersøkelsen gjennomføres eller ikke. Det ligger med andre ord et potensiale for å høyne kvaliteten på forskning ved hjelp av digitale tekster i det at tekstene blir søkbare.

Dette er nå en godt innarbeidet metode. Man digitaliserer tekster og gjør dem søkbare i fritekst, eventuelt med tilleggskriterier i søkingen. F.eks. kan man søke kun i innledninger, årstall, eller andre deler av teksten.[3] I denne oppgaven vil jeg undersøke andre former for bruk av tekster enn søking. Er det mer å hente i digitalisering av tekster enn bedret tilgang til faktiske tekststeder?

1.4.6 Presentasjon eller parasitt?

Maskinelle metoder kan benyttes til tekstanalyse, dels for å lage nye presentasjonsformer, men også for å generere helt nye tekster basert på den gamle. Disse nye tekstene kan f.eks. bestå av et sett av pekere i tillegg til en løpende tekst, og representere ny kunnskap lagt inn av menneske og maskin i samarbeid. Det er selvsagt slik at kunnskap lagt inn av maskinen er basert på modeller (programmer) som også er laget av mennesker, men for brukeren i prosessen oppleves det som om systemet man kjører på maskinen bidrar, til og med når man har skrevet programmet selv.

Det som skjer har trekk fra edisjonsfilologi og semantisk oppmerking av tekster, men er likevel noe annet — det er snakk om tekstkritisk arbeid for å generere kunnskap med utgangspunkt i den teksten som foreligger. Den nye kunnskapen henges på teksten, nesten som en parasitt eller som avleiringer utenpå grunnteksten. Det viktige her er at grunnteksten er ukrenkelig, den skal alltid kunne isoleres om man ønsker dét, men den skal også kunne flyte sammen med den fortolkende kunnskapen om man ønsker det. Man kan være forfatter, leser eller begge. Denne dobbeltheten er grunnleggende for denne oppgaven, og skal gjøre det lettere og bedre å bearbeide og vise fram det tillegget til grunnteksten som er den kunnskap forskeren har og det han tilegner seg i arbeidet. Denne kunn-

[3] Et eksempel på dette er søkesiden for Diplomatarium Norvegicum (DN-web), (Ore 2002).

skapen brukes dels til å legge inn nye opplysninger, dels til å se sammenhenger mellom elementer i originalen og uttrykke dem.

I slikt arbeide er det viktig at man er klar over hva man gjør. Noen former for oppmerking og beriking av en tekst kan brukes til å lage nye utgaver av den eksisterende teksten. Men annet arbeid vil uvergelig føre til at det lages en ny tekst — en avledet tekst i opphavsrettslig forstand.

I denne oppgaven vil jeg gjøre subjektive analyser av teksten. Jeg vil således gjøre oppmerking av teksten som er klart fortolkende, slik at grunntekst kombinert med mitt tillegg blir en avledet tekst. Analysene som gjøres skal være transparente, det betyr at det skal være åpenbart hvilke deler av den som kommer hvorfra: Hva er originaltekster, hva er regelbasert, hva er basert på kunnskap i forskererens hode. Dette tilsvarer en streng referansepolitikk man kjenner fra vitenskapelige tekster, men tilsvarer også de krav et ekspertsystem stiller til å i ettertid kunne forklare de slutninger som er gjort, jfr. avsnitt 1.2. Til dette bruker jeg det såkalte begrunnelsessystemet, som er beskrevet i avsnitt 2.5.

1.4.7 Bruk av referanser i denne oppgaven

I denne oppgaven lages det koblinger fra analyser til grunntekst gjennom en serie pekere.

Disse koblingene bidrar til å skille tekst og analyse. Dersom man i stedet hadde valgt å ekserpere fra teksten, ville det ekserperte ligge integrert inni analysene, og vil være tilpasset analysenes problemstillinger. I systemet i denne oppgaven er således grensene mellom grunntekst og analyse klarere definert enn ved en ekserpering, ved at teksten er knyttet til analysene gjennom et veldefinert grensesnitt.

Med utgangspunkt i samlingen av informasjon, jfr. avsnitt 1.5, kan således blikk som disse inn i materialet genereres:

1. Schnitler-utgave med kommentarer og hjelpetekster for person, sted, osv., og pekere ut til andre tekster.

2. En analyse av geografiske begreper med Schnitler som integrert kilde.

Valget mellom disse blir et spørsmål om hvor man ønsker å sette fokus. Beslektet med henvisninger er den tradisjonelle utgivelse av kildetekster. Den trykte Schnitler-utgaven og en vev- eller cdrom-utgave av den digitale versjonen er jo utgivelse av kildeteksster. Vi ser altså at både den klassiske kildehenvisningen og den klassiske kildetekstutgaven kan smelte inn i systemet jeg foreslår i denne oppgaven. Det finnes også i trykte utgaver grensetilfelle mellom en kommentert utgave av en kildetekst og en kritisk tekst med henvisninger til kildene — kommentarer knyttet til tekstutgaver kan gå langt i retning av en syntetisk framstilling av et sakskompleks basert på teksten, og motsatt, en tekst som utelukkende består av fotnoter som henviser til en og samme grunntekst kan være en måte å utgi denne kilden på (Grafton 1997, s. 120).

Det sentrale her er at disse ulike måtene å se teksten på kan *genereres* ut fra ett sett data. Viktig her er da å spesifisere hvordan dette kan gjøres. Dét er ikke tema for denne oppgaven, men et forslag til et system som gjør dette for arkeologisk materiale ble beskrevet på CAA-konferansen i 2004 (Eide 2004). Også prosjektet Henrik Ibsens skrifter baserer seg på et slikt tekstsyn: "Utgaven bygger på kontinental utgivelsesteori, og skal resultere i et digitalt tekstarkiv som kan danne utgangspunkt for publisering såvel i elektronisk form som i bokform." (Ystad 2000, s. 239). Man lager således ikke en utgave direkte, man bygger opp et arkiv på en slik måte at utgaver kan trekkes ut av dette.

Samtidig må vi huske at et åpenbart mål med Schnitlers arbeid fram mot manuskriptene var å skape en tekst som inneholdt viktig og riktig informasjon om geografiske forhold.[4] På samme måte som den trykte utgaven forsøkte å beholde teksten intakt for å bevare informasjonen, må digitale utgaver og annet digital bearbeidelse av teksten også respektere tekstens integritet.

1.5 Materialet

Schnitlers protokoller (Schnitler 1962) ble valgt ut som kilde for analysene i kapittel 3 fordi den i stor grad handler om geografiske forhold. Det er også en tekstsamling som finnes i digital form på en måte som egner seg for videre bearbeiding.

Den digitale teksten er basert på en trykt tekst, som igjen er basert på flere forelegg:

- En innledning skrevet av Kristian Nissen på 1950-tallet.
- Deler av Schnitlers Grenseeksaminasjonsprotokoller, transkripsjoner av et håndskrift fra 1740-tallet. Dette håndskriftet inneholder blant annet rettsprotokoller basert på avhør av vitner.
- Stedsregister utarbeidet av Ingolf Kvamen ca. 1960.
- Personregister utarbeidet av Ingolf Kvamen ca. 1960.

Denne teksten har vært gjennom en en digitalisering og SGML-tagging i 1995–2002, som jeg har vært ansvarlig for. For flere opplysninger, se sluttrapporten fra delprosjektet i Dokumentasjonsprosjektet (Eide 1998 A, særlig kapittel 2 og 3.).

1.5.1 Tekstenes tilblivelseshistorier

Disse ulike tekstene omtales her kort i kronologisk rekkefølge.

[4] Dette må selvsagt sees i kildekritisk lys av at oppdragsgiverne hans var en regjering med utenrikspolitiske mål.

Schnitlers protokoller

I forbindelse med forhandlingene fram mot grenseavtalen mellom Danmark-Norge og Sverige-Finland i 1751 (Strømstadtraktaten 1967) foretok major Peter Schnitler en grenseoppgang fra Røros til Øst-Finnmark der han organiserte rettslige avhør av befolkningen. Dette arbeidet er dokumentert i hans grenseeksaminasjonsprotokoller, hvis originaler finnes i Riksarkivet i Oslo sammen med kart, korrespondanse og annet materiale vedrørende hans arbeid med grenseproblematikken.

Protokollene ble ført i felten, og er en svært viktig kilde til forståelse av forholdene i grenseområdene i midten av det attende århundre, og inneholder også opplysninger om eldre historie. Selv om dokumentene må leses i lys av at de er skrevet av en dansk-norsk embetsmann med et bestemt oppdrag, inneholder de også viktig samisk kildemateriale, ikke minst som bakgrunn for den 1. kodisill som kom med i grenseavtalen av 1751, den såkalte "Lappekodicillen" (Eide 1998 A, ss. 20–21), (Hansen 2004, ss. 268 f.).

Selv om bakgrunnen for undersøkelsene er de aktuelle grenseforhandlingene, viser instruksene til Schnitler (Schnitler 1962, ss. XXVI f.) at undersøkelsene ble spesifisert i tråd med annen informasjonsinnhenting i Europa i denne perioden (Burke 2000, ss. 128 f.).

Protokollene består av 7 volumer. Av disse ble slutten av volum 2 og mesteparten av volum 3 og 4 sammen med noen andre dokumenter trykt som bind 1 av *Dokumenter angaaende flytlapperne* (Renbeitekommissionen 1909 2). Fra et restopplag av dette trykket fjernet man i 1926 de delene som ikke er Schnitlers protokoller, laget nytt tittelblad og utgav det som bind 2 i en samlet utgave av Schnitlers protokoller, *Major Peter Schnitlers grenseeksaminasjonsprotokoller 1742–1745* (Schnitler 1929). Bind 1 av samlingen, bestående av volum 1, de ikke trykte delene av volum 2–4 og volum 5 og 6, ble utgitt i 1962 (Schnitler 1962). De dekker Trøndelag, deler av Nordland, og Finnmark (ibid, ss. XXXIII–XL). Endelig ble volum 7 utgitt som bind 3 i 1985 (Schnitler 1985). Schnitlers tekst i det trykte bind 1 er på til sammen 435 sider og danner kildegrunnlaget for analysene i denne oppgaven.

Jeg har ikke selv gått gjennom håndskriftet for å se hvilke hender de ulike delene av denne utgaven er basert på, men bygger på Kristian Nissens oversikt (Schnitler 1962, ss. XXXIII–XXXIV) i tabell 1.1. Jeg kommer tilbake til dette i kapittel 3 fordi hvem som nedtegner teksten er en faktor som kan påvirke språklige formuleringer.

Nissens innledning

Innledningen er en sterkt forkortet utgave av Nissens opprinnelige manuskript, som var på omlag 250 sider. 38 sider fikk han på trykk (Schnitler 1962, s. VI). I innledningen gir han forhistorien fra 1720 til grensearbeidet på 1730- og 1740-tallet, skriver litt om Schnitler, og går gjennom arkivmaterialet i Riksarkivet. Innledningen blir ikke brukt direkte i denne oppgaven.

Volum	Hånd (vitneforklaringer)	Sider i bd. 1	Første node
1	P.J. Røyem	1–137	5 964
2	P.J. Røyem	138–178	41 503
3	P. Schnitler	179–184	53 029
4	P. Schnitler	185–206	54 584
5	P. Schnitler	207–415	61 816
6	Ukjent (etter P. Schnitler?)[5]	416–435	124 038

Tabell 1.1: Hender brukt i Schnitlers protokoller.

Kvammens registre

Ingolf Kvammen laget to registre, stedsnavnregisteret på 35 sider og personregisteret på 5 sider. Det siste av disse to brukes som tilleggskilde i analysene i denne oppgaven.

1.5.2 Digitale versjoner av tekstene

Selve teksten (bokstavene og andre tegn) i teksten er gjengitt diplomatisk i den digitale utgaven i henhold til den trykte utgaven. Den strukturelle og typografiske oppmerkingen er gjengitt i form av meta-tagger i henhold til SGML-standarden (ISO-8879). I tillegg er enkelte innholdstyper som ikke er konsekvent grafisk merket i den trykte teksten merket opp med SGML-tagger, så som steds- og personnavn. Mye av taggingen er således av deskriptiv art, men det er også en viss grad av fortolkende tagging.

Nissens innledning og Schnitlers protokoller

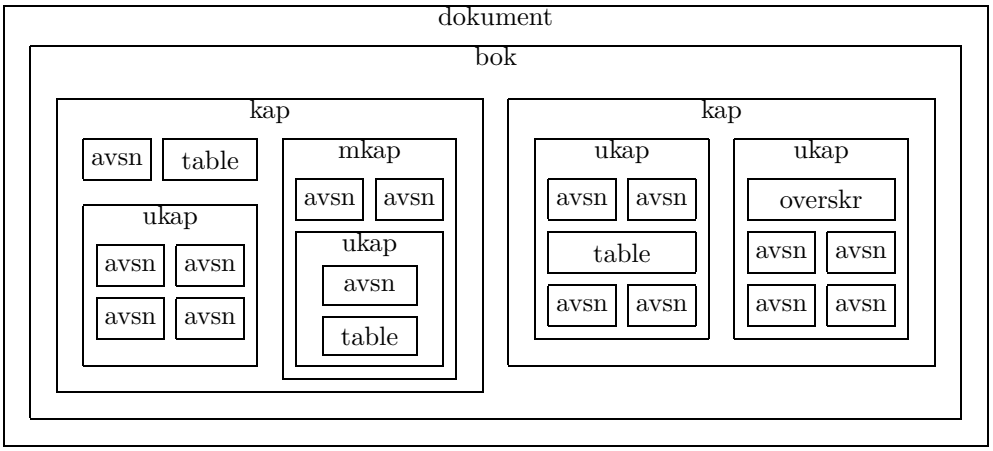
Teksten ble skannet, OCR-lest, korrekturlest og SGML-tagget ved Universitetenes registreringssentral i Indre Salten under veiledning av undertegnede.[6] Deretter gikk tekstene gjennom en sluttkorrektur i Tromsø utført av Nina Einevoll og undertegnede.[7] Den digitale teksten er en ren transkribering av den trykte utgaven, uten noen korrigering av trykkfeil. Schnitlers originale håndskrift er ikke konferert i arbeidet. En kan således si at den digitale teksten er en digital versjon av det trykte bindet, men SGML-taggingen gjør at det er lagt til informasjon som muliggjør ulike digitale utgaver. Informasjonssystemet beskrevet i denne oppgaven kan regnes som en slik digital utgave basert på deler av taggingen.

Databeskrivelsen til tekstene finnes i sluttrapporten fra delprosjektet i Dokumentasjonsprosjektet (Eide 1998 A, ss. 77–85). Jeg vil ikke gå gjennom den i detalj i denne oppgaven, men vil beskrive de viktigste elementene. Hovedstrukturen er grafisk framstilt i figur 1.1. Taggingen av teksten kan deles i tre

[5] "Ukjent" i volum 6 er antakelig kopi av en tapt original som ble ført av Schnitler.

[6] En grundigere gjennomgang av dette arbeidet finnes i sluttrapporten fra delprosjektet i Dokumentasjonsprosjektet (Eide 1998 A, ss. 25–47).

[7] Kvaliteten på teksten er drøftet sluttrapporten (ibid, s. 58).



Figur 1.1: Strukturen i dokumenter, gjengitt etter sluttrapporten fra delprosjektet i Dokumentasjonsprosjektet (Eide 1998 A, s. 34). Eksempler på strukturen innad i avsnittselementer finnes i figur 2.1 på side 27 og 2.2 på side 35.

hovedgrupper[8]:

Strukturelle elementer Kapitler, avsnitt, tabeller, etc.

Grafiske elementer Kursiv, halvfeit, midtstilt, etc.

Innholdsmessige elementer Dato, henvisninger, kilder, personnavn og stedsnavn

De første to gruppene er i liten grad fortolkende, mens den siste representerer i større grad en tolkning av teksten. Både ikke-fortolkende og fortolkende elementer danner grunnlag for analysene av tekstuniverset. I analysesystemet er også elementer som ikke var tagget i den digitale versjonen av tekstene lagt inn, og taggingen av eksisterende elementer er beriket.

Kvammens registre

Generering av stedsnavnregisteret er gjort på en annen måte enn resten av teksten, men resultatet har blitt en SGML-tekst tagget på samme måte som tekstdelene beskrevet ovenfor.

1. Teksten ble skrevet inn i Microsoft Word. Første del ble skrevet inn av Universitetenes registreringssentral i Indre Salten i Dokumentasjonsprosjektet, andre del av Javid Samadi i Museumsprosjektet. Strukturen ble markert med linjeskift, avsnittsskift og kursiv.

[8]Disse tre flyter i noen grad over i hverandre. F.eks. vises gjerne strukturelle elementer grafisk; ofte i manuskripter, og konsekvent i moderne trykte tekster. En overskrift er gjerne i større typer og midtstilt, et avsnitts første linje er gjerne rykket inn.

2. Strukturell tagging som direkte kunne utledes av Word-markeringen ble satt inn ved enkel søk-erstatt: Elementene `avsn` og `kur`.
3. Teksten ble autotagget i et Emacs-lisp-program.
4. Deretter ble taggingen korrekturlest manuelt.

1.6 Geografi midt på 1700-tallet

1.6.1 Lesekyndighet

Lesekyndigheten i det norske bondesamfunnet på 1740-tallet var preget av kirkelig undervisning, som også inkluderte en viss leseundervisning, og privatundervisning (Vannebo 1984, s. 5). Selv om en form for organisert skolegang ble innført i 1739, ble det gjort unntak for Nordlands og Finnmarkens amt. I 1775 var ennå ikke organisert skolevirksomhet satt i gang i dette området (Sogner 1996, s. 228). Men det var et viktig unntak fra dette.

Når man skal vurdere lesekyndigheten i Nord-Norge på denne tiden må man også se på spesielle tiltak rettet mot den samiske befolkningen. I de samiske samfunnene hadde det normale kirkelige systemet en viss innflytelse, både det dansk-norske, det svenske og det russiske. Men i tillegg kom den undervisningen som følger av den statlige finnemisjonen, som for alvor startet i 1716, altså 60 år før et organisert skolesystem ble innført for nordmenn i Nord-Norge.[9] Den førte til at

> flere av samedistriktene faktisk [fikk] et organisert skolestell før de første norske skoleforordninger (rundt 1740) ble iverksatt. På 1700-tallet viser da også visitasberetningene at "opplysningen" (særlig leseferdighet) jevnt over sto høyere blant samene enn blant nordmennene i Finnmark. (NOU 1985:14, s. 46).

Sannsynligvis var forholdene også gode på svensk side av grensen (Hansen 2004, s. 337).

I følge en undersøkelse fra Øst-Finnmark misjonsdistrikt satt opp av Gerhard Sandberg i 1775 var det kun 4–5 % av befolkningen som ikke kunne lese (Steen 1954, ss. 295–96), (NOU 1985:14, s. 47). En må ta slike opplysninger med en klype kildekritikk, da det er vanskelig å vite i ettertid hva som lå i "å lese", og leseferdigheter er vanskelige å måle (Vannebo 1984, s. 13, jfr. ss. 20–27). Det er ikke lett å oppdage om en person kan en tekst utenat eller om vedkommende leser den. Men uansett feilkilder er det lite som tyder på at flertallet av samer på 1740-tallet var skriftløse.

[9] Adolf Steen gir grundig gjennomgang av misjonshistorien, og i kap. 3 gjennomgås eldre kilder (Steen 1954). En nyere skildring som går nærmere inn på det politiske spillet i København er skrevet av Peggy Granås (Granaas 2002). Begge disse skildringene er promisjonske i den forstand at de ikke stiller spørsmålstegn ved at et religionsskifte var riktig og viktig. For forfattere som framstiller samisk før-kristen religion mer nyansert, se f.eks. Brita Pollan (Pollan 1993) og May-Liss Myrhaug (Myrhaug 1997). Nyere skildringer av misjonshistorien er skrevet av Håkan Rydving (Rydving 1995) og Lars-Ivar Hansen et.al. (Hansen 2004, ss. 327–37).

1.6.2 Skrivekyndighet

Selv om mange samer kunne lese rundt midten av 1700-tallet, var ikke den samiske kulturen skriftproduserende, noe den norske bondebefolkningen i Midt- og Norge-Norge som Schnitler forhørte heller ikke var i særlig grad. Det er viktig å være klar over at å kunne lese ikke alltid medfører å kunne skrive. Selv om disse ferdighetene i dagens barneopplæring og alfabetiseringsprogrammer læres nokså parallelt, ser det ut til at leseferdigheten i flere i-land ble spredt langt tidligere enn skriveferdigheten (Vannebo 1984, s. 5). En ser også at skriveundervisning på 1700-tallet ikke var obligatorisk slik leseundervisning var, og i mange tilfelle ikke engang ble tilbudt (Sogner 1996, s. 229).

> Når disse ferdighetene kan utvikles ulikt i et samfunn, henger dette trolig først og fremst sammen med det som tidligere ble kalt bruksbehovet innenfor vedkommende samfunn. [...] Ser vi på det norske samfunnet, eksisterte det f.eks. på 1700-tallet et klart behov for å kunne lese skrift — enten dette gjaldt tekster av religiøs eller verdslig art (trykte postiller, katekisme; handskrevne kjøpe- og salgskontrakter o.l.). Behovet for sjøl å kunne skrive var derimot mindre — i alle fall for store deler av befolkningen — da dette for mange enten ble ivaretatt av spesialutdannede "skrivere" eller det innskrenket seg til det å kunne sette navnetrekket sitt under et brev, en kjøpe- eller en salgskontrakt. (Vannebo 1984, s. 7)

I dette perspektivet er det naturlig å tro at samer som ikke kunne norsk hadde enda mindre grunn til å lære seg å skrive enn norsktalende, da behovet for skriftlig kommunikasjon var størst når man forholdt seg til "øvrigheten", som stort sett var norsktalende.

Dette er en del av forklaringen på hvorfor samer ikke framstår som subjekter i skriftlige kilder fra midten av 1700-tallet, noe som medfører at det ikke er noen direkte kilder til hvordan personer i disse kulturene formulerte seg. En direkte kilde som fortsatt eksisterer i dag må være skrevet ned, og kan derved ikke direkte avspeile en kultur uten skriftproduksjon. Det gjør at man må bruke andre metoder for å avdekke hvordan man formulerer seg i slike kulturer.

En mulighet er da å studere sekundære skriftlige kilder. Da må man ta hensyn til at materialet er filtrert gjennom en nedtegning, i en situasjon der nedtegneren hadde sin egen agenda som kun er delvis kjent for oss.

Når det gjelder analyse av språklige forskjeller mellom grupper av nålevende mennesker, er det ikke naturlig å bruke en metode som den som skisseres her. Da har man et langt rikere datagrunnlag, ikke minst fordi man kan ta uttale med i bildet. Den rent tekstbaserte metoden blir først aktuell på grunn av informasjonsfattigdommen hva periodens språklige uttrykk angår.

Hvis man skal studere faktiske forhold er det viktig også å vurdere i hvilken grad primærkilden fører sekundærkilden bak lyset. Et eksempel på dette er at når man skal studere den gamle samiske religionen er misjonærers skrifter fra 1700-tallet et viktig kildemateriale. Men man må lese slike kilder både ut fra

nedtegnerens agenda og ut fra primærkildenes evne til å forlede nedtegneren, noe f.eks. Adolf Steen gjør i for liten grad, jfr. fotnote 9.

Når det gjelder stilhistorie, *hvordan* man sa det man sa i motsetning til *innholdet* i det man sa, er det i mindre grad snakk om forleding av nedtegneren fra primærkildens side, men det er en åpenbar risiko for at nedtegneren "fører over" til sitt språk. Om dette skjer i så stor grad at det nivellerer ut forskjeller mellom grupper av talere er tema for analysene i kapittel 3 i denne oppgaven.

1.6.3 Schnitler som kilde til vitnenes formuleringer

Det er åpenbart at Schnitlers tekst gir faktakunnskap om geografiske historiske forhold. Om disse ikke alltid er korrekte, holder de et høyt kvalitetsnivå. Men i hvilken grad utgjør Schnitlers protokoller en kilde som viser hva menneskene som ble forhørt virkelig sa om geografi og også hvordan de sa det? Den følgende diskusjonen gjelder først og fremst de delene av protokollene som utgjør forhørsprotokoller, da det er her ikke skriftproduserende personers muntlige uttrykksform kan skinne igjennom.

I Daniel L. Smails studie av notarius publicus i Marseille i middelalderen framgår det at selv om en profesjonell skriver har sitt foretrukne geografiske system, vil ofte kunden som kjøper tjenester av skriveren foretrekke et annet system. Kundens system vil kunne påvirke skriveren, slik at kundegrupper som i stor grad foretrekker andre systemer enn skriveren delvis får skriveren til å skrive i "sitt" system — men bare delvis:

> We must assume that the clients of the notaries, when first asked by the notary to give property sites for the purposes of property conveyance, chose to define these sites according to their own language of space. The diversity of vernacular linguistic cartography, at times, managed to push through whatever standard form the notaries might have been developing, because the set of extant notarial site clauses includes every possible template and all manner of toponymic styles. [...]
>
> All the same, fourteenth-century notaries had a clear preference for streets, and often translated the cartographic terminology of their clients into the language of the streets. (Smail 1999, s. 67–68)

Dette kan ikke direkte overføres fra Marseille i middelalderen til Nord-Norge på 1740-tallet. Det er blant annet en viktig forskjell i statusforhold: Skriverne i Marseille var en form for håndverkere som solgte sine tjenester til kunder. De fleste forhørte personene i Midt- og Nord-Norge tilhørte lavstatusgrupper som forklarte seg til en kongelig embetsmann. Selv om de skulle ønske å påvirke hvordan deres budskap ble skrevet ned, var det nok vanskelig å presse skriveren til å følge deres instrukser.

På den annen side er det ting ved selve forhørssituasjonen som taler i retning av at vitnenes utsagn er nøyaktig gjengitt. Selv om det enkelte vitnes forklaring kun er ett element i et datagrunnlag for en sammenfatning av grenseområdenes

historie, er en edsbunden forklaring en alvorlig sak som det er viktig å forholde seg korrekt til. Følgende eksempel viser at eden er mer enn et tomt rituale — når man allerede har avlagt en ed, gjør man det ikke på nytt:

> Viidner, Som Skulle afhøres, vare [3 vitner] For dennem blev Eedens forklaring af Lov: bogen oplæst og aflagde de 2de Sidst benæfnte deris *Corporlig* Eed; dend 1te fra *Qvælie* blef kun erindret, at Siige Sin Sandhed i Kraft af hands forrige aflagde Eed. (Schnitler 1962, s. 140, jfr. også s. 366).

Videre ser vi den offentlige karakteren til en edbunden uttalelse, man måtte unngå eden ved underhånden undersøkelser:

> Som *Ordren* lyder, at *informere* sig om de Russiske Grændser under Haanden, saa toeg man Vidnerne paa denne deres udsagn ej *formelig* i Ed, men betydede dennem at ud sige deres Sandhed derom Som de, naar paakræves med Eed kunde bekræfte; (ibid, s. 423)

Dette kan sees i sammenheng med andre tekster av lignende type. De deler av Schnitler som jeg vier spesiell oppmerksomhet, forhørene, er ført som protokoller fra tingretter. Dette gjør det naturlig å sammenligne med tingbøker. Gjennom det såkalte Tingbokprosjektet (Tingbok WWW) er det arbeidet mye med slike tekster. Tilsvarende arbeid er også gjort i utlandet.

Man regner gjerne tingbøker som gode kilder. Erling Sandmo skriver i sin hovedoppgave, der han bruker tingbøker fra sent på 1700-tallet som kilder: "Det at de tingbøkene denne oppgaven er fundert på er opphavelige, nesten stenografiske nedtegnelser av det som ble sagt på tinget, reduserer sannsynligheten for at innførslene har gjennomgått noen dyptgripende redigering drastisk." (Sandmo 1992, s. 19) og han konkluderer: "Det at tingbøkene later til å være skrevet ned raskt og fortløpende, og at teksten til en viss grad gjør det mulig å skille ut ulike synsvinkler, gir grunnlag for å anta at kildematerialet i vår sammenheng tilfredsstiller grunnleggende krav til språklig autensitet." (ibid, s. 20).

Tim Stretton skriver at tingbøker tradisjonelt har hatt høy status som kilder: "Short of travelling back in time armed with tape recorders or clipboards, legal records appear to represent the closest we can get to the 'words as they were spoken' and the 'authentic voices' of thousands if individuals." (Stretton, 1997, s. 16) Han advarer likevel mot for sterk tiltro til slike kilder: "[It] remains a question whether, in the formal context of a legal interrogation, people used their everyday language and revealed their inner-most thoughts." (ibid, s. 24), og fortsetter med å flere referanser til andre forskere som påpeker det sannsynlige i at et vitne snakker forhøreren etter munnen. (ibid, jfr. også fotnote 24, s. 22). Han påpeker også at skriverens innflytelse viser seg gjennom normalisering av dialekter og identiske svar fra ulike vitner,[10] "suggesting that scribes paid more attention to the substance of answers than to their wording." (ibid).

[10] I gjengivelsene av vitneutsagn i Schnitler brukes "gjentagelsestegn": Det angis eksplisitt at vitnet sa det samme som et tidligere vitne, jfr. avsnitt 3.2.2.

Selv om man ikke skal godta de mest euforiske beskrivelsene av tingbøkers verdi som kildemateriale, er det ikke usannsynlig at trekk fra vitnenes måte å uttrykke seg på overlever gjennom skriveren inn i protokollen.

1.7 Mål for oppgavens analyser

Tilblivelsehistorien til den digitale teksten vi har i dag kan sees på som et sett av transformasjoner. I Schnitlers protokoller finnes en rekke hendelser nedtegnet. Denne nedtegnelsen er den første transformasjonen av hendelsene. Disse hendelsene inkluderer som en viktig del det aktørene sa. I denne sammenheng er talehandlingene og de andre handlingene primære hendelser og alle senere transformasjoner (nedtegning, avskrift, trykk, digitalisering, tagging) sekundære hendelser som har endret den opprinnelige informasjonen i større eller mindre grad. Alt er basert på den primære hendelsen, og vi går ut fra at en del av den er bevart, ellers ville ikke dette vært kildemateriale i det hele tatt. Nøyaktig hvor mye som er bevart vet vi imidlertid ikke, fordi vi ikke kjenner informasjonsendringen som skjedde i den første transformasjonen fra hendelse til håndskrift. De øvrige transformasjonene har vi bedre oversikt over. Vi må gå ut fra at denne første transformasjonen endret både form og innhold på det som ble sagt i betydelig grad. Det er likevel grunn til å tro at det ble bevart deler av uttrykksmønsteret til talerne i forhørene.

I denne oppgaven vil jeg lage et informasjonssystem (kapittel 2) og bruke dette til analyser som tar sikte på å finne systematiske forskjeller mellom måten grupper av talere[11] snakker på (kapittel 3). Ved å bruke systemet til å skille mellom talerne, kan man gruppere de tekstbitene som representerer intervjuer etter standen til intervjuobjektet. Man kan skille ut grupper som sjøsamer, reindriftssamer, bønder, embetsmenn, og andre. Embetsmennene er jo de som fører pennen (Schnitler og Røyem) og er i liten grad gjengitt som talere i forhørssituasjoner, bortsett fra at de stiller spørsmål. Selv om andre kategorier er med i analysegrunnlaget, er det de tre intervjuobjektkategoriene bønder, sjøsamer og reindriftssamer som først og fremst sammenlignes i analysedelen.

Det er trekk ved tradisjonell samisk geografisk tenkning som skiller seg fra den tenkning om slike forhold man kjenner fra moderne kartbaserte vestlige samfunn. Harald Gaski skriver at "Samiske landskapsbeskrivelser kan fungere som kart, der topografi, geografi og veiledning om den beste ruten å følge er inkorporert i hverandre." (Gaski 1995, s. 59) og videre at "det var viktig å kunne begrepene for terrengformasjoner og topografi for å være i stand til å kunne beskrive et geografisk område så godt at det ville være mulig også for andre å finne fram uten kart." (ibid, s. 63). Det er altså grunn til å tro at spesielle trekk ved samisk geografisk forståelse vil bli formulert som svar på spørsmål om geografiske forhold, og man kan også tenke seg at slike spesielle trekk et stykke

[11] Talere er i vår sammenheng de personene hvis muntlige utsagn nedtegnes i forhørsprotokollene knyttet til navn, jfr. avsnitt 3.2.1. En taler er altså en navngitt kilde til deler av teksten. Slike personer framstår således som subjekter i teksten, om enn filtrert gjennom en skrivers penn (dvs. hode).

på vei blir ivaretatt av en tolk med godt kjennskap til samiske forhold.[12]

Spørsmålet som ønskes besvart er således: Kan det finnes systematiske forskjeller mellom formuleringene om geografiske forhold uttalt av personer tilknyttet de ulike kategoriene talere? Med andre ord, kan man se at de tekstene embetsmennene skrev basert på intervjuer med bønder skiller seg fra det de skrev basert på intervjuer med reindriftssamer eller sjøsamer? Hva består slike forskjeller i tilfelle i?

En del av det som undersøkes er således i hvilken grad forskjeller nivelleres gjennom nedtegning. Er forskjellene mellom individer er større enn forskjellen mellom grupper, dvs. er forskjellene mellom individer systematisk i forhold til hvilke grupper de tilhører? Hvis jeg finner slike systematiske forskjeller må jeg så vurdere om de faktisk skyldes forhold ved primærkildenes måte å formulere seg på. Og dersom det ikke finnes systematisk forskjeller mellom talerstemmer eller kategorier av talerstemmer, kan det bety at selve nedtegningen av teksten har nivellert ut slike forskjeller.

1.7.1 Teser

Basert på vurderingene av tingbøker som historiske kildemateriale, har jeg satt opp følgende teser som blir undersøkt i kapittel 3 i denne oppgaven:

1. De delene av den digitale teksten som består av forhørsprotokoller inneholder vesentlige spor etter skaperne av formuleringene (talerne), selv om teksten er skrevet ned av en annen.

2. De ulike talerne kan grupperes etter klasse og etnisitet. Forskjellene mellom gruppene er større enn forskjellen mellom individer. Denne grupperingen sammenfaller med forskjeller i språkføring som henger sammen med forskjeller i tankesett.

3. Disse forskjellene kan påvises gjennom statistiske analyser etter en oppstykking, omgruppering og sortering av tekstbiter.

En person, det vil si den delen av teksten personen er definert som taler til, er grunnenheten i analysene, og systematiske forskjeller vil kunne framkomme ved at personer tilhørende én og samme kategori grupperer seg sammen, adskilt fra personer som tilhører andre kategorier.

Dersom det skulle finnes slike systematisk forskjeller, vil det bety at materialets kildeverdi som beskrivelse av talemåter styrkes, slik at visse former for undersøkelser av geografisk tankesett med utgangspunkt i dette materialet i større grad vil kunne forsvares. Det er vanskelig å gjøre undersøkelser av geografisk tankesett basert på disse tekstene før man har skaffet seg noe mer kunnskap om deres kildeverdi for formålet. Undersøkelsen av gruppeforskjeller

[12] En av tolkene i forhørene som nevnes ved navn er Erik Helset (Schnitler 1962, s. 140 og flere steder). Han var opprinnelig Thomas von Westens tjener, men ble misjonær i Overhalla fra 1721 (Steen 1954, s. 120, 403) og tjente som misjonær der og i Vefsn fram til 1738.

er en god måte å undersøke kildeverdien, i den forstand at et positivt resultat sannsynliggjør sterkt at kildeverdien er til stede.

Et negativt svar vil ikke nødvendigvis bety at materialet ikke har verdi som kilde til vitnenes talemåter, men mangelen på positivt resultat taler jo mot en slik verdi. Det kan bety at personer i de tre gruppene hadde uttrykksformer om geografi hvis innbyrdes forskjeller overskred de systematiske forskjellene mellom gruppene, eller det kan bety at det fantes systematiske forskjeller som ble nivellert ut av skriverne.

Kapittel 2

Informasjonssystemet

Informasjonssystemet er laget i tradisjonen fra systemene beskrevet i avsnitt 1.3.3. Det er videre bygget opp slik at man både skal kunne analysere tekster og kunne følge pekere fra analysene tilbake til grunnteksten, enten direkte eller indirekte. Til disse pekerne har man et sett av regler som viser hvordan bearbeidingen har skjedd. Der det har vært interaksjon med en menneskelig bruker er dette dokumentert, og systemet gir mulighet til å legge inn annoteringer for å begrunne det valget som er gjort ("Hvorfor det?").

Det betyr at selv om systemet er et system for analysestøtte, er det også et system som etter at analyseprosessen er gjort, gir brukeren en dokumentasjon av prosessen som er gjennomført. Denne dokumentasjonen er på alle punkter knyttet til steder i teksten. Mer om denne dokumentasjonen (begrunnelsessystemet) i avsnitt 2.5.

At analysene på denne måten er koblet til grunnteksten åpner også for nye former for publisering av analyser. Mer om dette i avsnitt 4.3.

2.1 Hovedstrukturen i implementasjonen

Det som følger er en kort skisse av strukturen i programmet. En gjennomgang av en programkjøring med beskrivelser av hvordan de ulike funksjonene brukes finnes i kjøringseksempelet i tillegg B. I tillegg vises praktisk bruk i kapittel 3. Selve programkoden er i tillegg A.

Programsystemet er skrevet i LISP (GNU CLISP 2001). Det parserer SGML-dokumentet og lagrer tekstuniverset i en trestruktur i minnet under kjøring. For å uttrykke informasjonen og muliggjøre resonnementer gjøres SGML-teksten tilgjengelig i en trestruktur og aksesseres fra programmet dels ved vanlige funksjoner for å navigere i trær (foreldre, søsken, barn), dels ved spesielle indekser. Systemet har således en Document Object Model-lignende arkitektur, men er ikke en implementasjon av W3C-standarden (DOM 2004).

Programsystemet består av flere pakker. Den ene (sgml-tre, se avsnitt 2.2 og tillegg A.1) er innenfor sine begrensninger generell for SGML-dokumenter

og inneholder ting som kan løses generelt for alle databeskrivelser (DTD'er). I visse tilfelle har jeg valgt å gjøre ting spesielt for å spare arbeid, men det sies da hvordan ting kunne gjøres generelt.

De andre pakkene er spesielle for de dokumenttypene vi jobber med, som er basert på en databeskrivelse kalt DOKUB (Eide 1998 A, ss. 77–85). De inneholder ting som ikke kan være generelle, blant annet fordi de baserer seg på begreper som side og navn som ikke finnes i alle SGML-dokumenter. Selv om disse funksjonene kunne vært skrevet mer generelt (f.eks. med parametriserte navn for sideskille, sidetall og navn), ville de ikke vært helt generelle. Dette gjelder parse-dokub, se avsnitt 2.3 og tillegg A.2, og pakken som implementerer metodene for dataanalyse, se avsnitt 2.4 og tillegg A.3.

En egen begrunnelsespakke lagrer begrunnelser som er knyttet til en del av de reglene som er implementert (grunner, se avsnitt 2.5 og tillegg A.4). Det skrives til denne regelbasen fra funksjoner i andre pakker. Det finnes metoder for å skrive ut de beslutningene som er tatt, både av systemet og av brukeren. Dette kan så brukes til å dokumentere de resultater man kommer fram til.

I tillegg finnes det en pakke med hjelpefunksjoner, herunder lagring og tilbakehenting av datastrukturer. Koden til denne finnes i tillegg A.5.

Beskrivelsen av systemet som følger er sentrert rundt en gjennomgang av reglene for transformasjoner som programmet implementerer med pekere til aktuelle funksjoner. Tranformasjoner er i denne forbindelse de endringer som gjøres med SGML-dokumentene når de gjøres om til data, og endringer i de dataene som gjør dem mer brukbare for formålet. Slike transformasjoner kan i større og mindre grad være fortolkende. Serien med transformasjoner er selvsagt lengre en dette — den starter på 1740-tallet og fortsetter forhåpentligvis inn i framtida.

Det er snakk om regler på to nivåer. Systemet som sådant definerer et sett av generelle regler. I tillegg brukes begrunnelsessystemet til spesielle begrunnelser. Det er et visst overlapp, men i hovedsak skal de generelle reglene være et underlag for de spesielle.

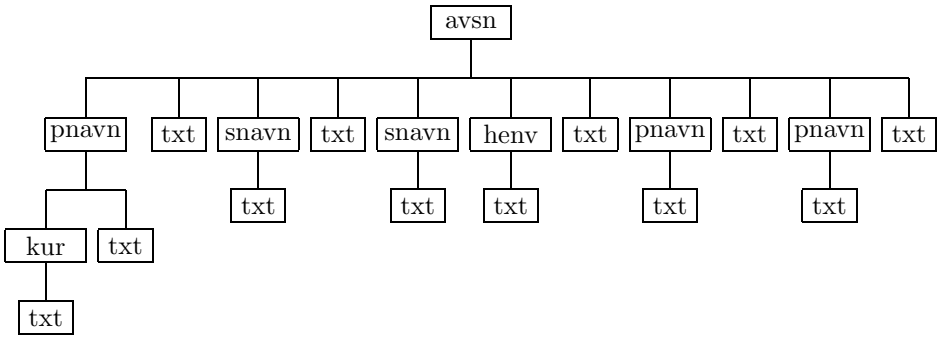
Jeg har delvis laget et system basert på menyvalg for handlingene, delvis et system der jeg har kalt rutiner fra lisp-promptet og skrevet nye rutiner ved behov.

2.2 SGML-pakke

SGML er en standard som åpner for mange ulike måter å kode dokumenter på, og det finnes ikke noe programsystem som implementerer SGML-standarden fullt ut. I dette systemet har jeg kun implementert de delene som har vært nødvendige for arbeidet.

2.2.1 Behandling av SGML

Basismodulen i programsystemet er et sett av metoder for å behandle SGML-baserte trestrukturer. En tekst leses inn, elementer og `PCDATA` isoleres og treet



Figur 2.1: Eksempel på SGML-fragment slik det er lagret i informasjonssystemet, uttrykt som et subtre.

bygges opp av objekter i en DOM-lignende struktur i minnet.

For å få dette til, leses SGML-teksten inn i en trestruktur i minnet, hvor hver node er representert ved et objekt som har metoder for å vise sin plass i treet, i tillegg til annen informasjon, som selve teksten den inneholder (for `PCDATA`-noder) eller attributtenes verdier (for `ELEMENT`-noder).

Et eksempel på hva som skjer i denne prosessen kan være følgende avsnitt fra personnavnregisteret:

> *Aanud* (Aanet, Aanod, Anud) Andersen, sjøfinn i Repparfjord,
> lagr.mann i Hammerfest pgld 241
> f. — Curtelius, svensk lappeprest,
> se Curtelius. (Schnitler 1962, s. 474)

Dette avsnittet ser slik ut i SGML-tagget versjon:

> <avsn><pnavn><kur>Aanud</kur> (Aanet, Aanod, Anud) Andersen</pnavn>, sjøfinn i <snavn>Repparfjord</snavn>,
> lagr.mann i <snavn>Hammerfest pgld</snavn> <henv>241
> f</henv>. <pnavn>— Curtelius</pnavn>, svensk lappeprest,
> se <pnavn>Curtelius</pnavn>.</avsn>

Dette avsnittet vil i informasjonssystemet representeres med et subtre som det i figur 2.1.

Det er en forutsetning at dokumentene er normalisert. Det betyr at ingen sluttagger er utelatt. Dette gjør at man i parseringen ikke behøver å sjekke navnet på sluttagger, fordi et element enten er tomt (contents model "empty") og ikke har sluttagg, eller også nester korrekt.

Entiteter behandles ikke spesielt i dette systemet, de sees på som en samling PCDATA-tegn. Dette er i og for seg greit for "character entities", som er den eneste typen entiteter i det aktuelle dokumentet, fordi de overlever inn og ut av systemet, men det betyr at det ikke finnes noen enkel metode for å erstatte entitetene med de verdier som skal settes inn for dem.

2.2.2 Utvidelse av opprinnelig tagging

Taggingen som ble gjort da den digitale teksten ble etablert (jfr. avsnitt 1.5.2) hadde ikke noe spesifikt siktemål (Eide 1998 A, særlig s. 45). I oppbyggingen av dette systemet er det derfor interessant å se hvilke deler av den opprinnelige taggingen som er nyttig, og hva som mangler. Vi ser at innholdstaggingen som ble gjort er nyttig. Særlig blir elementene `pnavn` og `snavn` brukt mye. Men det er også elementer vi kunne trenge som mangler.

Registeroppslag

I registrene var det ikke noe begrep om oppslag. For personregisteret trenger jeg dette når jeg skal koble personnavn i selve teksten til en enhet som inneholder kategori på personen. Vi har således laget et nytt element `pnavnoppsl` som omslutter alt som har med én person å gjøre. Jeg har også laget rutiner for å sette inn taggene automatisk, se avsnitt 2.3.2.

Taler

Alle deler av teksten bør om mulig ha angitt hvem som er det talende subjekt. I min sammenheng er det ikke nødvendigvis den som skriver, i forhørene setter jeg inn den forhørte.

Jeg har lagt inn mulighet for å legge inn taler i alle nodeobjekter. En taler er i vår sammenheng et personobjekt. Den ligger inne som et felt i nodeobjekter, i praksis på noen av `avsn`-nodene. Informasjonen legges inn manuelt med støtte fra systemet, se avsnitt 2.3.3.

2.2.3 Kryssende elementer

Et spesielt problem når man koder historiske kilder med SGML er hvordan systemet skal håndtere elementer som bryter med trestrukturen, såkalte overlappende elementer (TEI 2002, kap. 31). Metodene som er brukt her er dels fragmentering, dels milepæler, som beskrevet i (Eide, 1998, s. 45–47). I denne etterbruken av materialet tar jeg ikke spesielt hensyn til fragmenterte tagger, da fragmenteringen er gjort på grafiske elementer, ikke på innholdselementer (ibid, s. 45–46), slik at elementer jeg tillegger betydning i analysene ikke er fragmentert.

Milepæler, derimot, må behandles. De er som brytere — det vil si at en milepæl uttrykker at "noe" starter på det punktet. Hva "noe" er, vil spesifiseres her. Følgende milepæler finnes i databeskrivelsen som er brukt til denne teksten:

side sideskille

snr viser ved attributtet `nr` hvilken side vi er på. Denne siden varer fra forrige `side`-milepæl til neste.

spalter viser at teksten heretter var satt med det antall spalter som vist i `sp`-attributtet. Tekst før første `spalter`-element regnes for å ha `sp`-verdien 1.

spalte spalteskille

Spalteinformasjonen er ikke av spesiell interesse i dette arbeidet. Sideinformasjonen er derimot av stor betydning, da personregisterets innførsler peker til sidetall, jfr. avsnitt 2.3.1. Sidetall legges inn på alle noderobjekter, slik at alle bladnoder har lagret hvilken side de starter på, med unntak av side-bladnoder.

2.2.4 Avvik fra SGML-standarden

I programsystemet utviklet i forbindelse med denne oppgaven er ikke parseringen basert på databeskrivelsen (DTD). SGML-pakka baserer seg således på at dokumentet allerede er validert, hvilket det aktuelle dokumentet er. Det betyr også at en type informasjon som kun finnes i databeskrivelsen, nemlig hvilke elementer som er tomme, må finnes på andre måter. Når programmet ikke leser databeskrivelsen, er det ingen generell måte å sjekke om et element er definert som tomt (contents model `EMPTY`) på i SGML.[1] Selv om dokumentet er normalisert slik at minimaliserte elementer har fått inn sluttagg, slik de dokumentene vi leser inn er, er det ikke tillatt å sette inn sluttagg på elementer som er definert som `EMPTY` i databeskrivelsen.

Dette løses i vår versjon ved at man sjekker mot ei liste med tomme elementer som er hentet inn manuelt fra databeskrivelsen dokumentene er basert på og lagt inn i den global konstanten `*tomme-elementer*`.

Programmene tar heller ikke hensyn til at blanke og linjeskift utenfor steder som er definert til å kunne inneholde `PCDATA`, f.eks. mellom `</avsn>` og `<avsn>`, kan fjernes, men legger slikt inn som om det var `PCDATA`. Databeskrivelsen er nødvendig for å skille mellom `PCDATA` og hvit støy som kan oversees.

2.2.5 Transformasjonsregler

I arbeidet med å bygge opp SGML-representasjonen i minnet, er det implementert en rekke regler. Disse er generelle og er derfor ikke spesielt dokumentert i begrunnelsessystemet. Reglene som gjennomgås her er åpenbare konsekvenser av at man leser eller skriver SGML-dokumenter, og har ikke noe i et begrunnelsessystem å gjøre. Men de er en nødvendig del av dokumentasjonen av datasystemet, da de viser ting som f.eks. hvordan parseringen av dokumenter gjøres.

SGML-pakka tilbyr brukeren to hovedfunksjoner: `legg-inn-ny-sgml` (se kode på side 94), som bygger opp en trestruktur i minnet basert på SGML-strukturen i dokumentet, og `skriv-tre-til-fil` (se kode på side 101), som

[1] I XML har man et begrep om velformede dokumenter som skal kunne parseres uten databeskrivelse. Der løses problemet med tomme elementer ved at alle tomme elementer har avvikende syntaks, og at det ikke finnes tagg-minimalisering (XML 2004).

skriver ut det SGML-baserte treet. Pakka tilbyr også en rekke metoder for å bearbeide treet.

Ytterst i SGML-treet ligger en dummy rotnode. Dette gjør det mulig å legge flere SGML-fragmenter ved siden av hverandre, da rotnoden i hvert fragment blir barn av dummy-noden. Hvert fragment blir således et subtre av treet i minnet. Dette brukes til å legge selve Schnitler-teksten og personregisteret i minnet samtidig selv om de leses fra to ulike SGML-filer.

SGML-fila leses som en tekststrøm som hentes inn via et lesebuffer (`*buffer*`). Kjernen i innlesningen er gjenkjenning av tagger. Følgende syntaks gjenkjennes:

Tagger

- En tagg starter med "<" og slutter med ">".
- En sluttagg starter med "</".
- En teknisk tagg starter med "<!".[2]
- Alle tagger som ikke er sluttagger eller tekniske tagger er starttagger eller tomme element-tagger.

Attributter

- Dersom det kommer en blank i en starttagg innleder dette en rekke av attributter.
- Et attributt inneholder et navn, "=" og en verdi. Dersom verdien inneholder blanke, omsluttes den av " eller ′.

Elementer

- Et tomt element består av en tagg som representerer et tomt element.
- Et ikketomt element består av en starttagg, `PCDATA` og/eller andre elementer og en sluttagg.[3]

PCDATA

`PCDATA` er en strøm av tegn som ikke inneholder tagger. Siden linjeskift ble beholdt i den originale teksten, betyr det at delte ord ikke er satt sammen i SGML-teksten, og dette gjøres heller ikke ved import. Imidlertid gjøres det noen steder i analysesystemet.

[2] Dette er en upresis term som i våre filer inkluderer kommentarer og typedeklarasjon.
[3] Fordi SGML-dokumentene våre er normalisert, er dette alltid sant.

Operasjon 1: Les en SGML-tekst inn i minnet

Denne beskrivelsen viser de reglene som brukes i innlesingen av SGML-dokumentet. `*denne*` er en peker til aktuell node i treet.

2.1 *Elementnavnet finnes ved å lese fra "<" fram til første forekomst av ">" eller blank.*

2.2 *Attributter finnes ved å lese fra blank til blank i taggen. Til venstre for "=" står attributtnavnet. Til høyre står verdien. Dersom verdien innledes og avsluttes med enten enkle eller doble anførselstegn, regnes ikke blanke tegn som skilletegn. Attributtverdier omsluttet av enkle anførselstegn kan inneholde doble anførselstegn, men ikke enkle; tilsvarende for doble anførselstegn.*

2.3 *Hver gang en tagg leses, leses mellomrommet mellom foregående ">" og den "<" som innleder taggen. Dette er* `PCDATA`*, så dersom det er en ikke-tom streng, legges innholdet inn i en* `PCDATA`*-node som legges som siste barn til* `*denne*`*. Deretter:*

2.4 *Les fra "<" til første ">". Dersom andre tegn er "!", hopp over det leste og gå tilbake til 2.3 med resten av innstrømmen som argument. Hvis ikke:*

2.5 *Les fra "<" til første ">". Dersom andre tegn er "/", flytt* `*denne*` *til* `*denne*`*s mamma og gå tilbake til 2.3 med resten av innstrømmen som argument. Hvis ikke:*

2.6 *Les fra "<" til første ">". Dersom andre tegn ikke er "!" eller "/", finn elementnavnet (se regel 2.1). Dersom elementnavnet ikke finnes i lista over tomme elementer, opprett et nytt element som siste barn til* `*denne*`*, attributtene (se regel 2.2) legges inn i dette elementet, og* `*denne*` *settes til å peke på det nye elementet. Gå tilbake til 2.3 med resten av innstrømmen som argument. Hvis ikke:*

2.7 *Les fra "<" til første ">". Dersom andre tegn ikke er "!" eller "/", finn elementnavnet (se regel 2.1). Dersom elementnavnet finnes i lista over tomme elementer, opprettes et nytt element som siste barn til* `*denne*` *og attributtene (se regel 2.2) legges inn i dette elementet. Gå tilbake til 2.3 med resten av innstrømmen som argument.*

Operasjon 2: Skriv et SGML-tre fra minnet til en SGML-fil

For hvert av barna til det dummy rot-elementet følges reglene nedenfor:

2.8 *Dersom noden representerer et element:*

- *Skriv ut* ``<[elementnavn][attributtliste]>'' *(se regel 2.10).*
- *Skriv ut hvert av nodens barn.*

- *Dersom elementetnavnet representerer et tomt element, er utskrift av noden ferdig. Hvis ikke: Skriv ut* ``</[elementnavn]>''.

2.9 *Dersom noden representerer PCDATA, skriv ut PCDATA-tekststrengen.*

2.10 *For hvert attributt i attributtlista, skriv ut:* ``[blank][attributtnavn] = [skilletegn][attributtverdi][skilletegn]''. *Dersom* attributtverdi *inneholder dobbelt anførselstegn, brukes enkelt anførselstegn som skilletegn; hvis ikke, brukes doble.*

Dersom innlesing og utskrift fungerer, skal man kunne gjøre en nulltransformasjon og få samme fil ut som man startet med, dvs. `(skriv-tre-som-SGML (bygg-tre-av-SGML txt.sgml))`. Dette fungerer for vårt dokument, jfr. tillegg B.2.4.

Metoder for å arbeide på SGML-treet

En rekke metoder for å bearbeide og lese fra treet finnes. Av disse kan nevnes følgende:

legg-til-neste-barn (tre elementnavn attrib-liste type) Legger til en node sist av nodebarna under `tre`.

sett-inn-node (ny-node ny-mamma forste-barn siste-barn) Setter inn en ny node under `ny-mamma`. `forste-barn` og `siste-barn` brukes til å plukke hvilke noder som skal være barn av `ny-node` og hvilke som skal være søsken.

finn-pcdata-fra-tre (nodeliste) Returnerer en tekst som er konkatineringen av alle pcdata-tekstene i subtrærne med rotnoder i `nodeliste`.

finn-pcdata-fra-til (&optional (fra 1) (til *idnummer*)) Returnerer en tekst som er konkatineringen av alle pcdata i nodene med id fra `fra` til `til`.

finn-pcdata-ord-fra-noder (fra til) Skriver en liste av alle ordene i pcdata-noder i avsn-subtrær i nodene med id fra `fra` til `til`. Skalerer.

finn-pcdatanode-med-txt (nodeliste tekst) Returnerer første node i subtrærne i `nodeliste` som inneholder `tekst`.

sosken-rett-etter-eller-over (node stoppnavn) Returnerer første søsken rett etter `node` eller, hvis ingen søsken etter, søsken etter forfedre. Går ikke videre mot rota i treet når den kommer til første node som representerer elementer av typen `stoppnavn`.

For implementasjonsdetaljer og flere funksjoner, se kodelisting i tillegg A.1.

2.3 DOKUB-pakke

Dette er ei pakke som dels utvider den DOM-lignende strukturen i minnet som representerer SGML-dokumentene som ble lastet inn, og dels lager andre strukturer som kobles til den DOM-lignende.

For å gjøre dette enklere og mer effektivt, lagres noe informasjon i nodeobjektene basert på strukturen i dokumenter av vår type. Dessuten lagres den unike ID'en som hver node får som en attributtverdi, slik at den følger med ved en SGML-eksport av dokumentet.

I denne pakka ligger også menysystemet og kommandoene som laster de andre pakkene. Det er således denne pakka som lastes inn når programmet skal kjøres.

2.3.1 Sidetall

Våre dokumenter har et sidebegrep som er viktig fordi referansene fra registrene peker til sider. Hver nodes sidetall lagres derfor i nodeobjektet sammen med tallsystemet — tekstsamlingen har to pagineringer, en i romertall og en i arabiske tall.

Strukturen i side-taggingen i SGML-fila er:

- Sider er ikke tagget, da dette ville ført til overlappende elementer. I stedet er sideskiller tagget med det tomme elementet `side`.
- Sidetall er tagget med det tomme elementet `snr` der sidetallet (i romertall eller arabiske tall) finnes i attributtet `nr`.

Reglene for å tilordne sidetall til nodeobjektene er:

2.11 *Alle nodeobjekter hvis elementnavn er* `snr` *får sidetall hentet fra attributtet* `nr`*.*

2.12 *Alle nodeobjekter som representerer noder før første* `side`*-element får følgende sidetall, der* `s` *er sidetallet til første* `snr`*-element og* `ant` *er antall* `side`*-elementer før første* `snr`*-element:* $\texttt{s} - \texttt{ant}$.[4]

2.13 *For alle nodeobjekter som representerer noder mellom to* `side`*-noder eller mellom siste* `side`*-node og siste node: Dersom det finnes en og bare en node som representerer et* `snr`*-element, gis alle disse nodeobjektene denne nodens sidetall. Dersom ingen slike finnes, settes ikke sidetall.*[5] *Dersom mer enn en finnes, signaliseres en feil.*[6]

Side-tagger gis rimeligvis ikke noe sidetall. Sidetall vil vise sidetallet til elementets start, selv om elementet går over flere sider.

[4] Merk at dersom det finnes et `snr`-element før første `side`-element blir `ant` lik null og nodeobjektene får `snr`-elementets sidetall.

[5] Det hadde ikke vært noe problem å legge inn sidetall på lignende måte som for regel 2.12, men dette har vist seg ikke å være nødvendig i vårt tilfelle.

[6] Denne feilen oppstår aldri for vårt dokument slik det er nå, men i det opprinnelige dokumentet var det en feiltagging; en side-tag manglet. Dette ble oppdaget fordi det førte til to snr-tagger mellom to side-tagger.

2.3.2 Oppbygging av personnavnsystem

I analysedelen av denne oppgaven settes de enkelte personers uttrykksform i sammenheng med hvilken kategori de tilhører i henhold til en kategorisering som er basert på titler angitt i personnavnregisteret. Det betyr at det må lages et system som gjør det mulig å knytte en person både til vedkommendes kategori og til de avsnitt vedkommende kan sies å være ansvarlig for.

Når jeg her snakker om "person", er det den enheten i teksten som er knyttet til et bestemt sett av opplysninger. Det er ikke kun personnavn, for man kan ha ulike personer med samme navn. En person i mitt system er det som personnavnregisteret er definert som en personinnførsel, med navn og ofte tittel og andre opplysninger.

For å gjøre denne koblingen har vi et godt hjelpemiddel i personnavnregisteret. Dette inneholder ikke bare sidenummeret hver person omtales på, men angir også personens yrke eller stand der dette er mulig. Programmets opplisting av disse titlene sortert på kategorier jeg har gruppert dem inn i er brukt til å generere tabell 3.1 på side 47.

En viktig oppgave for denne pakka er således å bygge opp en navnestruktur og derigjennom lage koblinger mellom den delen av teksten som representerer 1700-tallsteksten og den som representerer personnavnregisteret fra 1960-tallet.

For å ta vare på opplysningene om personene er det opprettet en personnavnklasse som kalles `pnavn` og som inneholder det aktuelle navnet, avvikende navneformer, `pnavn`-elementer i registeret og i hovedteksten som er knyttet til navnet, sidehenvisninger, tittel, kategori og avsnittene navnet er knyttet til som taler.

"Tagging" av oppslag

Basisen for navnestrukturene er registrene. Hvis vi studerer eksempelet på side 27, ser vi at det ikke er merket i taggingen hva som er et oppslag. Selv om personnavnene er tagget, er det ikke noe i SGML-versjonen som angir at avsnittet inneholder to registeroppslag. Dette ser vi også på treet i figur 2.1 på side 27.

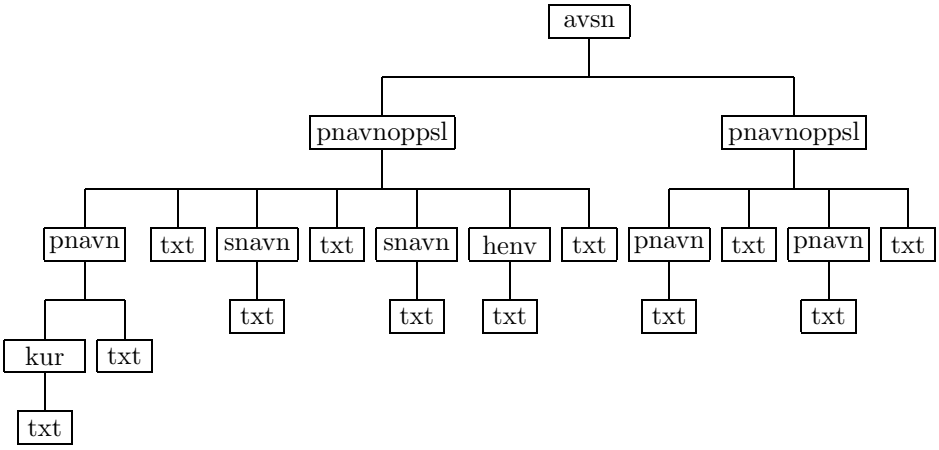
Det er imidlertid mulig å uttrykke mekaniske regler for dette. Reglene for hvilke pnavn-elementer som representerer starten på et oppslag er uttrykt i funksjonen `pnavn-oppslagp`, gjengitt på side 111, og inneholder:

2.14 `pnavn`*-elementer der det finnes et* `kur`*-element i subtreet under* `pnavn` *innleder et oppslag.*

2.15 `pnavn`*-elementer hvis* `pcdata` *innledes med en eller flere tankestreker innleder et oppslag.*

2.16 *Et oppslag varer til slutten av avsnittet eller til neste oppslag, det som kommer først.*

Hvis vi går tilbake til eksempelet på side 27, ser vi at det der må inn to `pnavnoppsl`-noder, slik at vi får følgende SGML-fragment, gjengitt grafisk i figur 2.2:



Figur 2.2: Eksempel på SGML-fragment slik det er lagret i informasjonssystemet etter tillegg av pnavnoppsl, uttrykt som et subtre.

> <avsn><pnavnoppsl><pnavn><kur>Aanud</kur> (Aanet, Aanod, Anud) Andersen</pnavn>, sjøfinn i <snavn>Repparfjord</snavn>, lagr.mann i <snavn>Hammerfest pgld</snavn><henv>241 f</henv>. </pnavnoppsl><pnavnoppsl><pnavn>— Curtelius</pnavn>, svensk lappeprest, se <pnavn>Curtelius</pnavn>.</pnavnoppsl></avsn>

Lista `*alle-oppslags-pnavn*` inneholder alle `pnavn`-noder som innleder et oppslag. Den lages av funksjonen `sett-alle-oppslags-pnavn` på side 111, som er en implementasjon av reglene 2.14 og 2.15. Funksjonen `sett-inn-pnavn-oppslag-node` på side 111 bruker denne lista og har i tillegg implementert regel 2.16 slik at den vet hvor oppslaget skal slutte. Den bruker så funksjonen `sett-inn-node` på side 93 til å gjøre de nødvendige endringer i den DOM-lignende strukturen i minnet.

Oppbygging av navneformer

For å gjøre det mulig å koble navnene i registeret til navnene i teksten, noe som gjøres ved å koble navneobjektene, som har en kobling til `pnavnoppsl`-noden fra de opprettes, til `pnavn`-nodene i Schnitler-teksten, er det nødvendig å legge inn flere alternative navneformer. I navneobjektet lagres navneformer to steder: Hovednavneformen lagres som `navn`, mens `navneformer` inneholder ei liste med alle de alternative formene av navnet. Det som skjer er altså en form for caching av former avledet etter ett sett regler ved at alle aktuelle muligheter listes opp. Etter at de mekaniske rutinene har gjort sitt, tilbys brukeren å korrigere resultatet i de tilfellene systemets regler definerer det som at det er tvil om resultatet er korrekt. Eksempler på utskrift av navneobjekter finnes i kjøringseksempelet i tillegg B.3.3. Innledningsvis er følgende aspekter ved oppsettet verdt å merke seg:

- Navnene er oppført med "etternavn, fornavn" for embetsmenn og andre personer av høyere stand og med "fornavn etternavn" for bønder, samer og andre laverestands personer. Det er se-henvisninger fra fornavn til etternavn for embetsmenn. Eksempler:
  - *Ahnen*, Preben von, amtmann i Nordlands amt 178.
  - *Ammond* Nielsen, fjellfinn, arrestant i Kjelvik 269.
  - *Preben* von Ahnen, amtmann, se A.
- En tankestrek først i et navn betyr gjentakelse av fornavn. Flere tankestreker betyr gjentakelse av flere navn. Eksempler:
  - *Anders* Ammondsen d. e., fjellfinn i Kautokeino, senere sjøfinn i Revsbotn 267 f. 410. — — d. y., sjøfinn i Revsbotn 268, 410. — Aslaksen, fjellfinn i Arisby 370. — Erichsen Breede, lensmann i Snåsa 97. — —, lagr.mann i Alta 240.
- Varierende navneformer settes typisk opp skilt med komma og/eller parentes. Eksempel:
  - *Aanud* (Aanet, Aanod, Anud) Andersen, sjøfinn i Repparfjord, lagr.mann i Hammerfest pgld 241 f.

Rutinene nedenfor brukes til å bygge opp de alternative navneformene i `navneformer`, med unntak av ekspandering av tankestrekene, hvor korrigert form også legges inn som hovedformen i `navn`.[7] De ulike endringene fører til at nye navneformer legges til ved behov: Først ekspanderes parentesene, deretter erstattes tankestrekene med riktig navneform, og til slutt legges en snudd navneform inn. Deretter kommer eventuelt interaksjonen med brukeren.

Snuing av navn med etternavn først
Selv om embetsmennene er angitt med etternavnet først i registeret, skrives de normalt med fornavnet først i teksten. Ved sammenligning mellom navneformer i registeret og navneformer i `pnavn`-noder i teksten er det derfor nyttig å kunne sammenligne navnet skrevet riktig vei, og uten komma. Navnet i snudd form legges derfor inn i lista over de alternative navneformene. Jeg har ikke implementert spesiell behandling av se-henvisningene.

Analyse av parenteser
Den aktuelle funksjonen er `slaa-sammen-med-parantes` som er gjengitt på side 107. Den er rekursiv og får inn navnet som ei liste av strenger delt på blanke, som dette: (s1 s2 s3 s4). Det betyr at navnet i eksempelet ovenfor kommer inn til funksjonen slik:

[7] Formen med tankestreker tas kun vare på i den forstand at det er en peker fra navneobjektet til nodeobjektet som utgjør roten til subtreet til teksten i oppslaget.

2.17 `'(``Aanud'' ``(Aanet,'' ``Aanod,'' ``Anud)'' ``Andersen'')`

Ved påfølgende kall brukes tre optional-parametre:

status som kan ha følgende verdier:

par en parentes er startet, men ikke avsluttet

ikke-par ingen aktiv parentes

utliste som samler opp de ferdige navnene i ei liste

utstreng som samler opp det som skal bli ett navn i en streng

Reglene er som følger:

2.18 *Hvis* `s1` *er tom: Ferdig — returnerer* `utliste` *reversert.*

2.19 *Hvis* `s2` *er tom:*

- *Hvis* `status` *er* `par` *(vi er inne i parentes): Ferdig — konkatinerer* `s1` *med* `utstreng`*, legger det på* `utliste`*, og returnerer denne reversert.*
- *Hvis ikke: Ferdig — legger* `s1` *på* `utliste` *og returnerer denne reversert.*

2.20 *Hvis* `status` *er* `par` *(vi er inne i parentes):*

- *Hvis siste tegn i* `s1` *er ")" (vi avslutter parentes): konkatinerer* `s1` *med* `utstreng`*, legger det på* `utliste`*, og kaller rekursivt med lista som starter på* `s2`*,* `status` *satt til* `ikke-par` *og* `utstreng` *tom.*
- *Hvis ikke: Kaller rekursivt med* `s1` *konkatinert på* `utstreng`*.*

2.21 *Hvis siste tegn i* `s1` *ikke er "," og første tegn i* `s2` *er "(" (vi starter ny parentes i den forstand at foregående skal kunne erstattes. Dersom forrige ble avsluttet med komma er det en annen type parentes):*

- *Hvis siste tegn i* `s2` *er ")" (vi avslutter parentesen med en gang):* `s1` *og* `s2` *konkatineres og legges på* `utliste`*, og kaller rekursivt med lista som starter på* `s3`*.*
- *Hvis ikke: Kaller rekursivt med* `status` *satt til* `par` *og s1 lagt i* `utstreng`*.*

2.22 *Hvis ingenting som har med parentes å gjøre, kalles rekursivt med* `s1` *lagt på* `utliste`*.*

Mer kompliserte eksempler, f.eks. nestede parenteser og parentes kombinert med snudd navn, takles ikke av denne funksjonen. Slike kompliserte oppsett er uansett så sjeldne at det er mer rasjonelt at brukeren hjelper til med å løse problemene.

Analyse av tankestreker
Oppsett med `pnavn` som inneholdet tankestreker baserer seg på en lineær form typisk for trykte dokumenter. Hver tankestrek betyr gjentakelse av en del av forrige navn.[8] I et system der den lineære strukturen brytes ned, må tankestrekene erstattes med korrekt navneform. Det gjøres ved at prosedyren `kompletter-pnavn`, gjengitt på side 108, husker navnet på forrige objekt i form av ei liste av strenger. Dette er implementert ved hjelp av en vektor, `*forrige-liste*`, som initielt er tom. For hver navnestreng i lista med navnestrenger som utgjør et fullt navn følges følgende hovedregler:

2.23 *Dersom ord nummer* `n` *i lista med navnestrenger er —, erstattes ord nummer* `n` *i lista med navnestrenger med ord nummer* `n` *i* `*forrige-liste*`.

2.24 *Dersom ord nummer* `n` *i lista med navnestrenger ikke er —, erstattes ord nummer* `n` *i* `*forrige-liste*` *med ord nummer* `n` *i lista med navnestrenger.*

Interaksjon med brukeren
Interaksjonen styres på makronivå av et flagg som kan skrus av eller på og som styrer om det i det hele tatt stilles spørsmål til brukeren. Hvis det er skrudd av, vil resultatet som kommer ut være det beste funksjonene får til uten hjelp av brukeren. Dersom interaksjonen er skrudd på, vil kompliserte oppsett bli forelagt brukeren til godkjenning og eventuelt endring.

Interaksjon brukes på to steder i forbindelse med navnearbeidet, og flagget kan godt være på i det ene og av i det andre. Først tilbys brukeren å være med å bygge opp et tittel-kategoriregister, samtidig som hovedformen til navnet sjekkes. Deretter kan brukeren delta i arbeidet med å lage de alternative navneformene, dvs. at når de alternative navneformene er generert, gis brukeren mulighet til å korrigere resultatet.

Dersom man ikke gir manuell hjelp i oppbygging av tittel-kategori-registeret, vil ikke noen kategorier tilordnes, fordi systemet ikke har noen predefinerte kategorier. Men register-pnavnnoder i teksten med og uten interaksjon i generering av navneformer vil fungere brukbart fordi systemet klarer å legge inn korrekte navneformer på mange navn.

Kategorilista og ei liste med pekere fra navneform til de avledede formene kan også lastes inn fra fil. Dersom listene som lastes inn er komplette, vil ikke interaksjon være aktuelt i slike tilfelle. Man kan også manipulere de lagrede filene, selv om det ikke er laget noe godt verktøy for å gjøre det. Det betyr at man f.eks. kan fjerne former man ikke liker, laste inn data på nytt og kjøre rutinene om igjen. Da vil kun de man fjernet bli laget på nytt.

Interaksjonen (se tillegg B.3.1 og B.3.2) fungerer slik at ved innlegging av 443 navneobjekter opprettes et tittel-kategoriregister like mange innførsler. Til dette brukes 152 interaksjoner, men av disse interaksjonene er det kun 32 som innebærer innskriving av nye former. De fleste interaksjonene innebærer kun å velge tall. Det betyr at alle korrekte former er listet opp, men at det gjerne er

[8] I de SGML-baserte tekstene er tankestrekene kodet med entiteten `—`, men jeg gjengir dem som — her for å gjøre det mer oversiktlig.

feilaktige former i tillegg. Det fører til at formene man trenger for å koble navnene ville vært funnet også uten interaksjon på 120 av de interaktive tilfellene. Formene som forkastes manuelt er stort sett ikke realistiske og vil derfor ikke føre til falske positive.

Kobling mellom navneobjekter og nodeobjekter av typen pnavn

Soundex

Soundex er en algoritme som brukes mye i arbeid med persondata. I den kodes navn på en måte som skal gjøre at avvikende stavinger av samme navn skal bli likt kodet, mens ulike navn kodes ulikt (NARA 2000). Et slikt mål er selvsagt umulig å nå fullt ut.

Soundex er tilpasset engelsk språk. Det er laget norske tilpasninger, blant annet ble det laget for bruk i Dokumentasjonsprosjektet.[9] Ved en gjennomgang av det aktuelle navnematerialet, er det klart at en rekke endringer til soundex er aktuelle for vårt danskspråklige materiale. Imidlertid viser det seg at tilnærmingen fungerer godt nok for vårt formål uten endringer.

Bruk av Soundex innebærer at navneformene regnes som like dersom en tilnærming av formene er like. Det betyr nødvendigvis at det er en viss andel falske positive. Det er imidlertid ikke så mange at det skaper store problemer for oss.

Koblingsprosessen Selve koblingen mellom navneobjekt og `pnavn`-node skjer i funksjonen `legg-pnavn-paa-pnavnreg` (gjengitt på side 115). Den får inn et personobjekt og ei side der det kan være forekomster av dette navnet. Den finner alle `pnavn`-nodene på sida og sammenligner teksten i hvert `pnavn`-element med hver av navneformene i navneobjektet, inkludert de avvikende navneformene, ved hjelp av soundex-koding av hvert navn. Soundex-sammenligningen skjer i hjelpefunksjonen `finn-igjen-soundex` (gjengitt på side 167).

Når sammenligningen slår til, legges en peker til navneobjektet inn i `pnavn`-nodeobjektet, en peker til noden legges til lista over noder i navneobjektet, og en begrunnelse skrives.

I koblingsprosessen (menyvalg B2, kjøringseksempel i B.3.2) kalles `legg-pnavn-paa-pnavnreg` (gjengitt på side 115) med hver person og hver side som står oppgitt i henvisningen fra registeret for personen. En "f" etter et sidetall tolkes som siden og siden etter, mens en "ff" etter et sidetall tolkes som siden og fire sider etter.

Problemer i struktureringen

Å koble navn er lettere enn man skulle tro, for navneoppsettet idet intervjuet starter ligner gjerne på det i registeret. Det er også rimelig, siden registeret er basert på teksten. Det vil si at personnavn i registeret med lite informasjon eller mangelfulle navneformer sjelden representerer talere, selv om de av og til er ansvarlige for skriftlige tillegg.

[9] Lars-Jørgen Tvedt: Privat meddelelse.

Men et grunnleggende problem er at registre som dette er laget for å leses av mennesker. Det er vanlig med variasjon som av en leser knapt merkes, og i alle fall ikke oppfattes negativt, men som for et dataprogram lager store problemer. Det kan virke irriterende i dag, men er basert på en annen måte å forholde seg til tekster på enn ren syntaktisk parsering. Et eksempel på dette er at avvikende navneformer i mesteparten av registret settes opp som vist på side 36, men plutselig, mot slutten av registeret, kommer oppsett som dette:

Thomas, Thomes, Tomes, Andersen, fjellfinn i Overhalla pgld 160.

Dette er likt nok til at en menneskelig leser knapt ser forskjell, mens et dataprogram som kun følger en syntaktisk mal ikke klarer å takle det. De avvikende tilfellene er imidlertid så få at vi ikke lager rutiner for det, men lar brukeren ordne det manuelt. Det gjøres ved at hovednavneformen korrigeres til standard oppsett, i dette tilfellet "Thomas (Thomes, Tomes) Andersen". I slike systemer er det alltid en avveining hvor mye som skal automatiseres. Det er alltid en rest av tilfeller. Poenget er å gjøre 90% maskinelt, men så gjøre det enkelt å få de siste 10% inn i systemet gjennom manuell interaksjon.

2.3.3 Tilordning av taler til avsnitt

Etter at `pnavn`-elementene i teksten er koblet til personobjektene, trenger brukeren et verktøy som gjør det mulig å spesifisere hvilken person som er taler til de enkelte avsnittene. Dette kan ikke gjøres automatisk, men jeg har laget et halvautomatisk system der brukeren får hjelp i arbeidet. Se eksempel på bruk i kjøringseksempelet i tillegg B.4.

Når brukeren starter den interaktive prosedyren, skriver han inn et nodenummer. Da vises teksten i første avsnitt etter noden, sammen med ei liste over aktuelle kandidater til talerrollen. I den listen finnes følgende objekter i oppgitt rekkefølge:

1. Personobjektet som representerer taleren som ble tilordnet til forrige avsnitt
2. Første navn i lista med aktuelle kandidater til forrige avsnitt
3. Alle `pnavn`-nodeobjekter i dette avsnittet
4. Personobjektet som representerer den spesielle taleren "Uspesifisert skriver"
5. Personobjektet som representerer den spesielle taleren "Skal ikke analyseres"
6. Resten av personnavnene med aktuelle kandidater til forrige avsnitt

Lista inneholder således både `pnavn`-nodeobjekter og personnavnobjekter. Duplikater blir fjernet, og lista blir om nødvendig kuttet bakfra til ni innførsler.

Brukeren velger taler ved å trykke et tall. Dersom taleren ikke finnes i lista, kan han søke i navneobjektene.

Dersom brukeren velger en taler som ikke er representert i lista ved navneobjektet, brukes koblingen mellom `pnavn`-noden og navneobjektet (se ovenfor) til å få tak i navneobjektet. Avsnittets node legges til i lista over tilordnede talere i navneobjektet. Hvis koblingen mellom `pnavn`-noden og navneobjektet mangler, får brukeren mulighet til å søke etter personen i navneobjektene og velge riktig person. Da legges det samtidig inn en kobling fra `pnavn`-noden til navneobjektet, slik at problemet med manglende koblinger beskrevet ovenfor løses der det er behov for det for mitt formål.

I koblingen mellom registerets personnavn og `pnavn`-nodene er det et visst antall falske positive. For å unngå dette, kunne man la være å kjøre koblingen mellom `pnavn`-noder og personobjekter og la brukeren legge inn alle koblinger manuelt. I mitt tilfelle viste seg at det er et så lite antall av koblingene som ble brukt som var feilkoblet at det ikke ble noe problem.

Brukeren kan endre tilordning av taler til avsnitt (se kjøringseksempel i tillegg B.4.2), men det er ingen brukervennlig måte å endre koblinger mellom `pnavn`-noder og navneobjekter på.

2.4 Analysepakke

De pakkene som er beskrevet til nå bygger opp en datastruktur som er deskriptiv. Den lagrer data fra SGML-fila og utleder trekk som bedre og mer effektivt beskriver disse dataene, og brukeren kommer med påstander om sammenhenger mellom ulike deler. Nå går vi over til ei pakke som bruker denne deskriptive datastrukturen til å understøtte analyse av tekstene.

Analysepakka bygger opp et sett analysedata knyttet til noder i det DOM-lignende treet og til andre strukturer, som navneobjektene. Dette brukes så som grunnlag for prosedyrer som dels hjelper brukeren til å gjennomføre analyser, dels tilrettelegger data for analyse i eksterne programmet, som SPSS (SPSS 2003) og Lisp-stat (XLISP-STAT 1999), ved å eksportere data. Jeg inkluderer ikke metoder for statistikk eller grafisk visualisering i mitt system, da dette er kompliserte funksjoner som er implementert bedre i andre systemer enn jeg vil kunne få til.

De to hovedgruppene analyser som er implementert er frekvensanalyse og naboanalyse. I tillegg finnes noen rutiner for gruppering og visning av tekst.

2.4.1 Frekvensanalyse

Grunnstammen i verktøyene for frekvensanalyse er hashtabellen `*frekvenstabeller*` og noen hjelpetabeller. `*frekvenstabeller*` holder en tabell for hvert personobjekt som er spesifisert som taler til minst ett avsnitt. Disse tabellene inneholder samlede ordfrekvenser for alle ord i alle avsnittene personen er taler for. Før analyser kan gjøres, må denne tabellen populeres. Det gjøres i funksjonen `lag-leksikon-med-frekvens-alle-personer` (gjengitt på

side 138). For hver person henter funksjonen ut teksten fra alle avsnitt vedkommende er taler til. Ordene i denne teksten grupperes og telles, og resultatet legges inn i `*frekvenstabeller*` med personobjektets id som identifikator.

Når så de ulike analyseverktøyene skal brukes, leses data fra `*frekvenstabeller*` og brukes som grunnlag for analysene. Flere detaljer om hvordan dette gjøres finnes i beskrivelsen av frekvensanalysen i avsnitt 3.4 og knyttet til kjøringseksempelet i tillegg B.5.

2.4.2 Naboanalyse

Naboanalysene baserer seg på samme metodikk som frekvensanalysene. Hashtabellen heter her `*kontekst-etter-snavn-tabeller*`, og den populeres av funksjonen `lag-kontekst-etter-snavn-tabeller` (side 151). Merk at denne tabellen inneholder kontekst etter `snavn`-noder og ut avsnittet som strenger. Det betyr at analyse av teksten mellom to stedsnavn gjøres ved at man tar slike strenger fra starten fram til første `snavn`-node.

Når så de ulike analyseverktøyene skal brukes, leses data fra `*kontekst-etter-snavn-tabeller*` og brukes som grunnlag for analysene. Flere detaljer om hvordan dette gjøres finnes i beskrivelsen av naboanalysen i avsnitt 3.5 og knyttet til kjøringseksempelet i tillegg B.6.

2.4.3 Gruppering og visning av tekst

Ordlengder beregnes i funksjonen `snittlengde-ord` (side 148) ved at det lages ei ordliste basert på alle ord i et sett avsnitt (f.eks. en talers, en kategoris eller alle taleres), og lengden summeres og deles på antall ord. Eksempler på bruk i analysene i avsnitt 3.3 og knyttet til kjøringseksempelet i tillegg B.7.

KWIC-konkordanser av ulike slag genereres av funksjonen `KWIC-konk-nodeliste` (side 157). For hver konkordanslinje sendes nodenummeret til avsnittet med. Dette er ikke noen veldig god kobling tilbake til grunntekst, men man har i hvert fall mulighet til å nøste opp hvor de ulike linjene kommer fra. Eksempler på hvordan dette brukes finnes i analysene i kapittel 3 og knyttet til kjøringseksempelet i tillegg B.7.

2.5 System for begrunnelser

Med utgangspunkt i ideene fra kapittel 1 har jeg laget et system for digitale referanser som er knyttet til resten av systemet. Jeg kaller disse referansene for begrunnelser, fordi de brukes til å angi årsaken til at en viss datamengde er slik den er, det er snakk om begrunnelser i valgsituasjoner.

Begrunnelsessystemet er implementert som en datastruktur hvor objekter som representerer grunner kobles til de øvrige objektene i systemet.

Her vil jeg gå gjennom virkemåten til systemet, mens jeg i kjøringseksemplene i tillegg B viser hvordan begrunnelser er integrert i bruken av programmet. I tillegg brukes begrunnelsene direkte i analysen i avsnitt 3.5.3.

Begrunnelsessystemet definerer en egen klasse `grunner`. Denne inneholder opplysninger om en hendelse som har skjedd på et tidspunkt[10] og som påvirker andre objekter i systemet. Et `grunner`-objekt har en type, en årsak, en ansvarlig (som kan være en funksjon i programmet, evt. med et brukervalg og en begrunnelse for dette skrevet inn av brukeren), og kobling til andre hendelser. Konkrete eksempler på hvordan de ulike begrunnelsene er knyttet til andre objekter i systemet finnes knyttet til kjøringseksemplene i tillegg B.3.3 og B.8.

Begrunnelsesobjektene indekseres på de ulike objekttypenes id'er. Programmet har predefinert hva som begrunnes, men en tekst kan skrives inn i tillegg til informasjonen fra systemet i noen situasjoner. Hvorvidt dette gjøres er selvsagt opp til brukeren. Følgende avgjørelser i pakken `parse-dokub` er valgt å begrunnes:

Innførsel i navneregisteret (`lag-navnereg-innforsel` og `lag-navnereg`)

Innførsel i tittelregisteret (`finn-tittel`)

Kobling mellom pnavn-node og personobjekt (`legg-pnavn-paa-pnavnreg`)

Navneformer på et navneobjekt (`legg-inn-navneformer-paa-ett`)

Angi taler til et avsnitt (`velg-taler-til-avsnitt`)

Endre taler til et avsnitt (`endre-taler-til-avsnitt`)

Når det skal legges inn en ny begrunnelse et sted i systemet, kalles funksjonen `ny-grunn` (kode på side 161) som oppretter et grunn-objekt, og som legger inn riktige typer på de objektene grunnen skal kobles til. På side 65 er det et eksempel på hvordan informasjon lagt inn i begrunnelsessystemet viste seg å være til nytte i analysen.

[10] Tidspunktet er ikke lagret, se avsnitt 4.1.2.

Kapittel 3

Analyser av tekstuniverset

3.1 Bakgrunn

I forrige kapittel ble programsystemet for dataanalyse beskrevet. I dette kapitlet skal jeg vise et eksempel på bruk av systemet. Jeg går da tilbake til tesene formulert i avsnitt 1.7.1, og undersøker materialet for å bekrefte eller avkrefte disse.

Det er en grunnleggende forskjell mellom historisk tekstanalyse og tekstbaserte analyser man kjenner fra estetiske fag. En historisk analyse er orientert mot hendelser i større grad enn mot tekst, den baserer seg på et sett av ulike kilder som skal gi en syntetisk historie. Ved å vurdere kildene opp mot hverandre streber man etter å komme så nær som mulig en historisk sannhet, hva som virkelig skjedde.

I tekstanalyser slik de kjennes fra fag som litteraturvitenskap eller teologi leser man gjerne én tekst av gangen, og forsøker å finne ut hva akkurat denne teksten betyr. Andre tekster kommer inn som støttetekster, med en sekundær funksjon.

I denne oppgaven ligger metoden mer opp mot en litteraturvitenskaplig enn en historisk lesning. Selv om målet er å finne ut noe om historiske forhold, om hvordan ulike samfunnsgrupper på 1740-tallet tenkte om geografi, brukes ikke kildematerialet som grunnlag til å si noe om de fakta som omtales der. Spørsmålet er ikke hva som sies, men hvordan det sies.

Målet med analysene er altså å sannsynliggjøre at det finnes ulike grupper av talere som samsvarer med kategoriene bønder, sjøsamer og reindriftssamer og som det finnes stilistiske forskjeller mellom uttalelsene til.

I analysene studeres kun ordene i teksten, mens skilletegn utelates. Med ord mener jeg således samlinger av bokstaver og tall som står rett etter hverandre. Sammenslåingen av ord som i SGML-teksten er delt i orddeling er gjort mekanisk og er ikke alltid korrekt. I noen av analysene skiftes hele stedsnavn og personnavn ut med ett enkelt ord, henholdsvis SNAVN og PNAVN.

3.2 Oppdeling av tekstuniverset

3.2.1 Kategorier av talere

I registeret "Identificering av personnavne" (Schnitler 1962, s. 474–78) gis personene titler som jeg har lagt inn i personobjektene.[1] Disse har jeg gruppert videre i kategorier som vist i tabell 3.1. Denne grupperingen er gjort under oppbyggingen av navneregisteret, se avsnitt 2.3.2. Titlene er en del av informasjonene i det trykte registeret, med unntak av "Uspesifisert skriver", "Forhører" og "Skal ikke analyseres". Tabellen viser hvor mange ord i de delene av teksten som er tilordnet taler som er tilordnet talere i hver kategori. Tilordningen av tekst til taler har skjedd manuelt, se avsnitt 2.3.3. Av tabellen framgår det hvilke grupper i samfunnet som i størst grad framstår som kilder hos Schnitler — det er *ikke* overklassen — embetsmennene er stort sett Schnitler og hans medarbeidere selv.

3.2.2 Arbeidet med å inndele i kategorier

Avsnitt ble valgt som analyseenhet.[2] Lengden på hvert avsnitt er viktig for analyseverdien. Avsnitt på mindre enn to linjer inneholder stort sett faste formuleringer, gjentagelse, navn og lignende. Dessuten er ofte stemmen vekslende. Jeg legger derfor kun inn taler på avsnitt på hundre tegn eller mer. Dette sparer manuelt arbeid i koblingen, fordi brukeren ikke behøver å forholde seg til mange korte avsnitt som gir liten analysemessig uttelling.

Hele teksten ble gjennomgått i et interaktivt program som listet opp hvert avsnitt dersom det inneholdt mer enn 99 tegn og noen forslag til talere. Brukeren (som også er meg) valgte så en person som taler og la evt. inn en begrunnelse for valget, jfr. avsnitt 2.3.3 og kjøringseksempelet i tillegg B.4.

Noen observerte trekk ved avhørene

Forhørte personer har stort sett gode registerinnførsler, fordi protokollen oppgir navn og stand. Derimot har de som har undertegnet gamle vedlagte skrifter ofte kun et navn.

Første vitne på et sted "snakker" ofte mest, de andre er i større grad henvisninger til tidligere vitner. Man merker i tillegg en forskjell mellom taleføre menn og menn som svarer kortest mulig. Typisk oppsett er at første vitne på et sted gjengis utførlig, mens de andre kun gjengis der de avviker fra første.

> Til det 6^{te}: *Resp:* Det Samme Som første viidne.
>
> Til det 7: *Resp:* Svarer som første viidne undtagen der i, at hand icke Egentlig veed Tallet af opsidderne ved *Ferragen* Søe, og at der er eendeehl af dem som holder 1: hæst og Kiørøxene: (Schnitler 1962, s. 5)

[1] Dersom det finnes flere titler i et personoppslag velges normalt det første.

[2] Jeg har analysert innholdet i avsnitt og i overskrifter, men har utelatt tabeller.

Kategori	**Nr.**	**Ant. ord**	**Titler**
adel	1	575	Geheime Conferense Raad, dronning, konge, prins
arbeider	2	2 201	gruvearbeider
bonde	3	31 734	plassmann, nybygger, godseier, husmann, gardsarbeider, bumann, bonde
embedsmann	4	74 412	lensherherre, landshovding, tingskriver, sorenskriver, befalingsmann, foged, amtmann. I tillegg "uspesifisert skriver" og "forhører"
fastboende	5	0	bufast nordmann
funksjonær	6	177	direktør
håndverker	7	0	bergmeister
jeger	8	0	skogmann
kirke	9	352	erkebisp, misjonær, sokneprest, klokker, finnemisjonær, kapellan, resid. kannik, superintendent, prest, prost
kvæn	10	1281	kven
lagrett	11	837	lagmann, lagr.mann
offiser	12	10 735	regimentskvartermester, major, løytant, kaptein, oberstløytant, oberst, general, generalløytnant
politi	13	6 093	lensmannen, lappelensmann, bonde-lensmann, vicelensmannen, vicelensmann, finnelensmann, lensmann
reindriftssame	14	15 886	fjelllapp, flyttlapp, fjellapp, fjellap, finn, fjellfinn, lapp
sjøsame	15	22 646	bufinn, sjøfinn
skole	16	175	lector, misjons-skolemester, skolemester
soldat	17	3352	garnisonssoldat

Tabell 3.1: Alle kategoriene (menyvalg D9 og B4).

Det er umulig å vite sikkert om slike henvisninger til tidligere vitner er vitnets, forhørers eller protokollførers kobling, men det er vel sannsynlig at Schnitler gjør slike koblinger.

De ulike kategoriene

Det er tre kategorier som er opprettet utenfor det vanlige kategorisystemet. Disse tre kaller jeg dummy-personer. "Uspesifisert skriver" tilhører kategorien embedsmann og ble tilordnet alle avsnitt som det ikke var mulig å koble til en konkret skriver eller taler. Alle avsnitt tildeles "Uspesifisert skriver" dersom de ikke eksplisitt er gjengivelse av en eller flere navngitte personers tale eller skrift, eller dersom de ikke knyttes en av de to andre dummy-personene. "Forhører" tilhører kategorien embedsmann og brukes til spørsmålene som stilles i forhørene, mens "Skal ikke analyseres" tildeles avsnitt som jeg av ulike grunner ikke ønsker analysert. Det kan være alt fra avsnitt der spørsmål og svar er blandet til oppramsing av tall. Denne siste dummy-personen er ikke knyttet til noen kategori.

Det er ikke i alle tilfelle mulig å vite om beskrivende tekst er skrevet av Schnitler eller en av de andre i hans følge, men det er ikke så vesentlig for oss. Det viktigste er å dele inn personene som taler i grupper, så som embedsmann, bonde og reindriftssame.

"Uspesifisert skriver" er i praksis Røyem og Schnitler, jfr. avsnitt 1.5.1. Deler av teksten som er undertegnet Schnitler og der han bruker førstepersonsform er tilordnet ham, dette er litt inkonsekvent fordi lignende avsnitt med mindre eksplisitt skriver er tilordnet "Uspesifisert skriver". Det gir også et litt skjevt resultat fordi Schnitler er tilordnet kategorien offiser, ikke embedsmann. Men disse kategoriene sammenlignes i liten grad, så det ikke noe stort problem.

Årsakene til eventuelle forskjeller mellom bønder, samer og arbeidere på den ene siden og embetsmenn på den andre er vanskelig å fastslå fordi de første er forhørte og de andre har andre roller. Dette er mindre viktig i dette arbeidet fordi det er forholdet mellom bønder, sjøsamer og reindriftssamer som undersøkes, ikke forholdet mellom disse tre gruppene og embetsmenn.

Gruppeavhør

I noen av avhørene, særlig i volum V, forhøres grupper av vitner samlet. Disse gruppeforhørene passer ikke helt inn i vår måte å gruppere talere på. Jeg kunne lagt inn nye kategorier for grupper, men det viste seg at i de fleste tilfellene tilhører vitnene i gruppa samme kategori. Det gjorde det mulig å bruke en av personene som representant for gruppa, samtidig som en forklaring ble lagt inn i begrunnelsen for valget.[3] I noen tilfelle er dette vanskelig, også fordi forhørene glir sammen med skriverens oppsummering. I slike tilfelle brukes "Uspesifisert skriver" eller "Skal ikke analyseres".

[3]Dette viste seg i senere å være nyttig informasjon, jfr. side 65.

Kategori	*Antall personer*	*Gjennomsnitt*	*Median*	*Varians*
	1	3,6805	3,6805	.
adel	1	5,0760	5,0760	.
arbeider	2	4,3052	4,3052	0,014
bonde	34	4,3735	4,3371	0,038
embedsmann	7	4,7358	4,8641	0,136
funksjonær	1	4,3718	4,3718	.
kirke	1	5,4107	5,4107	.
kvæn	3	4,7795	4,3603	0,571
lagrett	1	4,3811	4,3811	.
NIL	4	4,5828	4,6145	0,122
offiser	1	4,7516	4,7516	.
politi	4	4,4194	4,4361	0,014
reindriftssame	31	4,4093	4,4292	0,042
sjøsame	22	4,3990	4,4431	0,050
skole	1	4,4405	4,4405	.
soldat	1	4,3906	4,3906	.
Total	115	4,4415	4,3906	0,081

Tabell 3.2: Ordlengde for alle personer ordnet etter kategori

Begrunnelser

Begrunnelsessystemet ("Hvorfor det?") kan blant annet brukes til forklaring der tilfellene er på grensen mellom flere kategorier og forhør av flere personer nedtegnes samlet. Muligheten for forskeren til å knytte inn fritekst i systemer som dette er viktig.

3.3 Ordlengde

I den første analysen så jeg etter systematiske forskjeller i ordlengde mellom personer i ulike kategorier.

Ordlengdene for alle personer ble målt (menyvalg D11, jfr. tillegg B.5.9) og listet opp sammen med kategori. Tabellen 3.2 viser de ulike kategorienes gjennomsnittslengde, median og varians. Dette ble beregnet i SPSS (SPSS 2003).

Resultatet viser ingen systematisk gruppering av personenes ordlengde inn i kategorier. Den eneste som skiller seg ut er "Skal ikke analyseres" (blank kategori), som har noe kortere ordlengde enn de øvrige. Dette skyldes at mange av tekstfragmentene som er klassifisert her er oppramsing av avstander o.l. "Adel" og "kirke" har lengre gjennomsnittlig ordlengde, dette skyldes at tekstene er formelle skriftlige henvendelser. Ellers ser vi at de kategoriene som har mer enn 10 000 ord (jfr. tabell 3.1) har svært like verdier.

Jeg forsøkte også å se på fordelingen av ord med ulike lengde for de ulike kategoriene, dvs. hvor stor andel av alle ordene som var på tre tegn, fire tegn

Ord	*ant*	*Ord*	*ant*	*Ord*	*ant*	*Ord*	*ant*	*Ord*	*ant*
SNAVN	13883	har	973	1/4	604	vester	352	land	283
og	6418	om	971	et	604	nu	350	finner	278
i	6366	denne	907	søer	549	østen	345	uden	277
til	3266	dend	868	lang	538	østre	344	bredt	271
fra	2933	nord	868	side	532	een	342	imod	268
er	2928	dette	836	imellem	525	4	340	botten	267
af	2873	sig	822	efter	512	svenske	339	deris	256
paa	2559	over	786	3	496	ende	335	væster	254
at	2254	PNAVN	785	field	489	disse	334	noget	251
miil	2168	hvor	774	her	485	breed	334	kan	251
det	2152	hand	761	j	475	siide	327	ikke	245
de	1972	1/2	747	være	443	søndre	323	man	236
som	1865	øster	736	ere	405	vestre	322	skal	235
den	1603	norske	733	næss	404	ad	310	norden	231
for	1593	eller	690	han	385	skoug	306	2de	230
med	1453	saa	641	dog	383	1/8	305	pag	229
en	1254	have	637	samme	378	bøsseskud	305	fielde	227
1	1048	2	620	nordre	366	aar	302	siden	223
ved	1041	ligger	619	miile	365	deres	288	naar	220
der	999	men	608	langt	362	sønden	284		

Tabell 3.3: De 99 mest frekvente ordformene.

osv. for de ulike personene (menyvalg D12, jfr. tillegg B.5.10). Dette gav heller ingen tendenser som grupperte personene i samsvar med kategoriene.

3.4 Ordfrekvenser

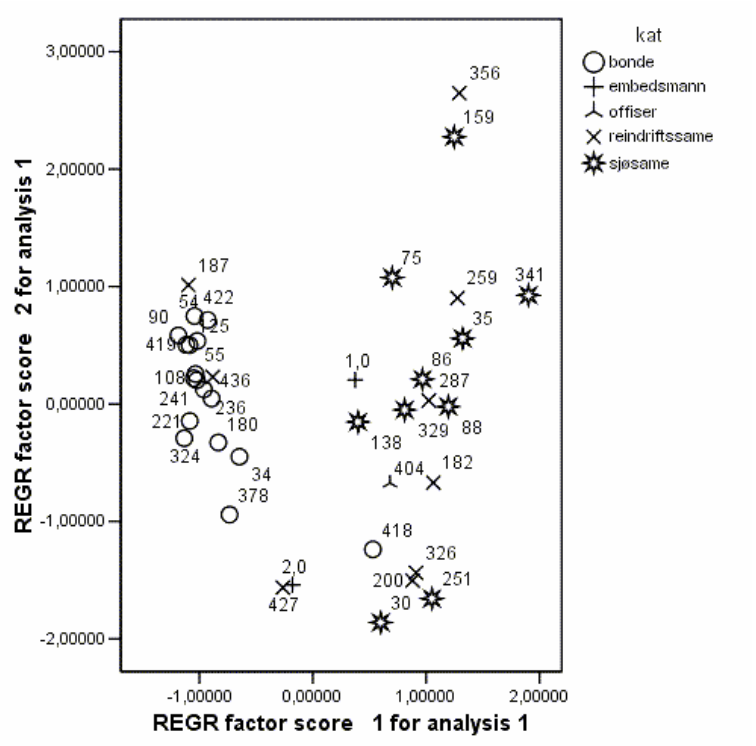
3.4.1 Ordformer og kategorier

De delene av teksten som er tildelt taler består av 170 916 ordformer. De 99 mest frekvente ordformene er listet opp i tabell 3.3. Avvikende stavemåter av samme ord er ikke slått sammen, men merk at ordformene skrevet med store bokstaver i tabellen representerer elementer, dvs. alle strenger som er tagget opp som hhv. stedsnavn og personnavn.

Kategoriene som velges ut for analyse er alle kategorier som har flere enn 10 000 ord samlet, jfr. tabell 3.1 på side 47. Dette er 3 (bonde), 4 (embedsmann), 12 (offiser), 14 (reindriftssame) og 15 (sjøsame). Kategori 4 og 12 er egentlig den samme, 12 (offiser) brukes på Schnitler når han eksplisitt er angitt som skriver ("jeg"), jfr. avsnitt 3.2.1. De kategoriene som i analysen vil sammenlignes for å finne forskjeller er bønder, sjøsamer og reindriftssamer.

3.4.2 Finnes systematiske forskjeller?

For de 99 ordene som var mest frekvente i hele det tildelte materialet ble frekvensene for alle talere tilhørende disse fem gruppene eksportert i et passende



Figur 3.1: Faktoranalyse av de 99 mest frekvente ordformene for alle talere med mer enn 500 ord i relevante kategorier.

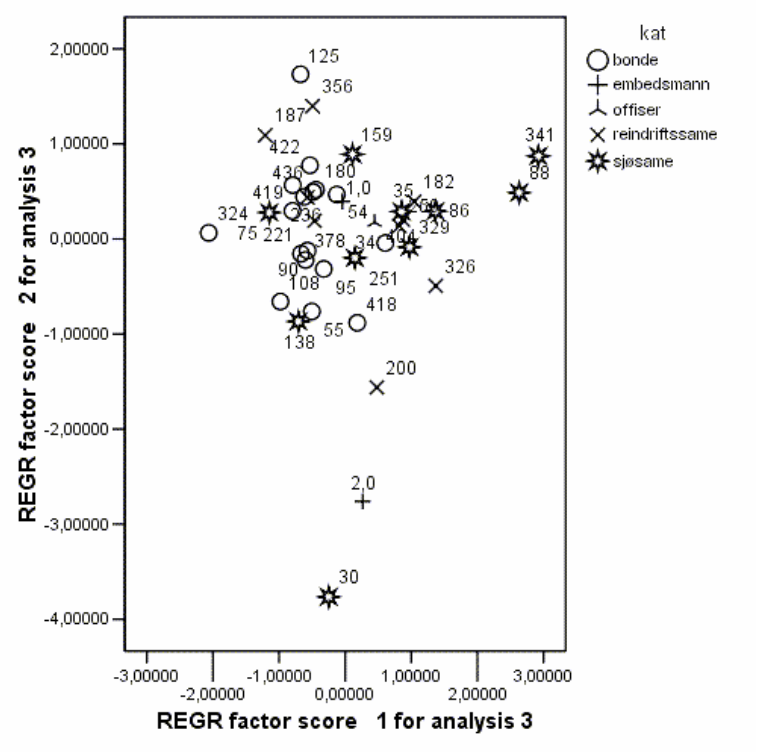
format (menyvalg D16, jfr. side 212) og importert i SPSS (SPSS 2003). Der kjørte jeg en faktoranalyse (Oakes 1998, s. 105 ff.) på alle 99 ordene.[4] De to første faktorene forklarte kun 21,2% av variansen — jeg måtte opp i 10 variable for å forklare over 50%. De to første faktorene ble så plottet i et diagram. Dette viste ingen klare tendenser, men en viss forskjell mellom de fleste bønder på den ene siden og reindriftssamer og sjøsamer på den andre.

For å unngå data med lite informasjonsverdi i analysen, eksporterte jeg på nytt, men utelot alle talere som ytret færre enn 500 ord. Ved samme analyse i SPSS, forklarte de to første faktorene 32,2% av variansen, og plottet (se figur 3.1) ble noe tydeligere.

Her er det åpenbart en viss samling av bønder. Jeg forsøkte å gå ned til å analyse kun de 20 mest frekvente ordene. Dette økte rimeligvis forklaringsverdien av de to første faktorene, til 40,8, men plottet var nærmest identisk.

Jeg har nå sannsynliggjort at det finnes en viss forskjell mellom gruppene.

[4]Menyvalg `Data Reduction` — `Factor`. Parametrene som ble brukt var `Variable`: alle 99 ordene. `Correlation Matrix`: `Cofficients`. `Extraction`: `Principal components`. `Analyze`: `Correlation matrix`. `Number of factors`: 2. `Rotation`: `None`. `Scores`: `Save as variables`.



Figur 3.2: Faktoranalyse av de 14 preposisjonene blant de 99 mest frekvente ordformene for alle talere med mer enn 500 ord i relevante kategorier.

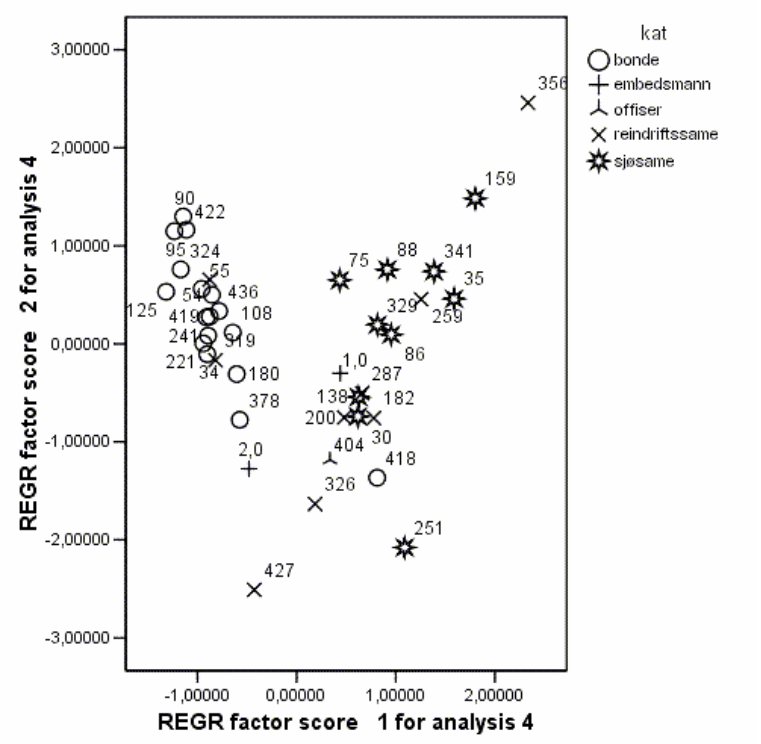
Men det betyr ikke at jeg har vist noen stilforskjell — forskjellene kan like gjerne være basert på forskjellige temaer i det personene snakker om.

3.4.3 Hva skyldes forskjellene?

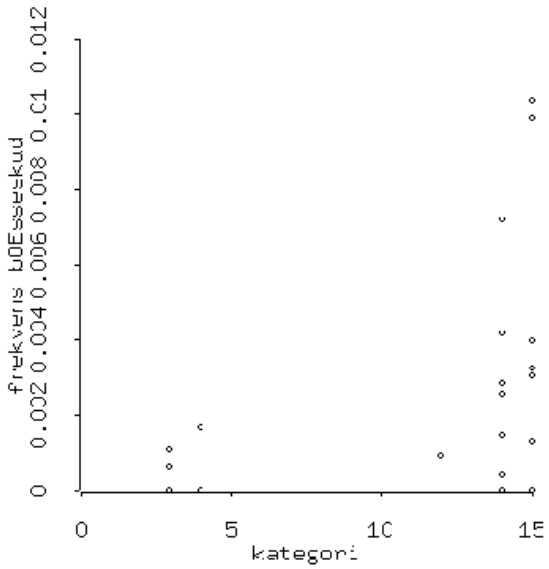
Jeg forsøkte to analyser til i SPSS for å forsøke å finne ut hva forskjellene kan komme av. Først valgte jeg ut ordformer som typisk brukes som preposisjoner og analyserte dem med de samme parametrene som ovenfor.[5] Dette er typiske funksjonsord som ofte brukes i stilanalyse. Det var 14 slike ordformer i materialet. De to første faktorene forklarte kun 33,2% av variansen, og plottet var kaotisk, ingen systematisk fordeling av talere med ulik kategori (figur 3.2).

Så gjorde jeg tilsvarende analyse for de 27 ordformene som typisk brukes som substantiver i materialet med de samme parametrene som ovenfor.[6] Her forklarte de to første faktorene 45,0% av variansen, og plottet lignet sterkt på plottet for alle 99 ordformene, se figur 3.3.

[5] Eneste unntak er at "Variable" kun var disse 14 ordformene.
[6] Eneste unntak er at "Variable" var disse 27 ordformene.



Figur 3.3: Faktoranalyse av de 27 substantivene blant de 99 mest frekvente ord for alle talere med mer enn 500 ord i relevante kategorier.

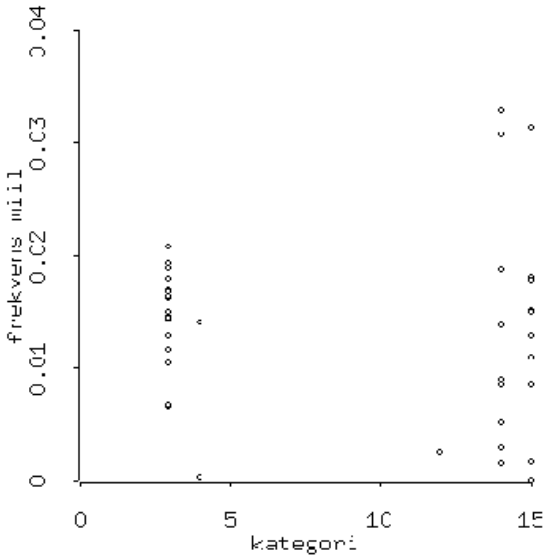


Figur 3.4: Plotting av *bøsseskud* mot kategori. I denne figuren, og de neste to, viser y-aksen andelen av ord som er det analyserte ordet, mens x-aksen viser kategoriene. Hver prikk representerer frekvensen til en eller flere talere. Kategorienes navn kan gjenfinnes i tabell 3.1 på side 47.

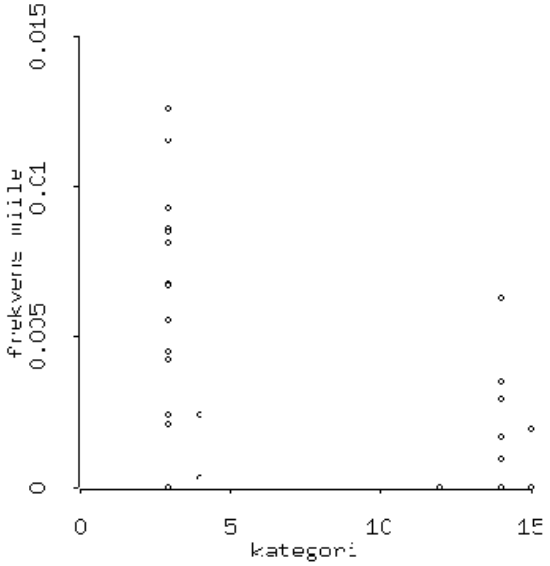
Dette at den systematiske variansen mellom kategorier er knyttet til substantiver (som i stor grad markerer tematikk) og ikke til preposisjoner (som i stor grad markerer stil), er en indikasjon på at forskjellene mellom kategoriene i hovedsak er en forskjell i tematikk. For å se nærmere på dette, laget jeg en eksport fra systemet til Lisp-stat (XLISP-STAT 1999) som viser fordelingen for de 99 mest frekvente ordene, ett og ett, fordelt på kategori, kun talere med minst 500 ord (menyvalg D16, jfr. side B.5.13)[7]. Dette brukte jeg til manuelt å velge ut de ordene som viste interessante trekk, og se hva slags ord dette var.

For disse kategoriene lages et plot i Lisp-stat for hvert ord med en prikk for hver taler. Et program som kan kjøres i Lisp-stat lages ved menyvalg D14, jfr. tillegg B.5.12. Alle talere veier således like mye i plottingen, uavhengig av antall ord de er ansvarlige for. Ved gjennomgang av disse plottene, plukkes ordene med klare forskjeller mellom kategoriene ut for videre analyse. Ordene som ble plukket var: *botten, bredt, bøsseskud, den, dend, deres, deris, een, han, hand, langt, miile, pag, side, siden, siide, sønden, søndre, vester, vestre, væster, østre.*

[7]Det var lite forskjell mellom lista som kun inneholdt talere med minst 500 ord og den med alle talere.



Figur 3.5: Plotting av *miil* mot kategori.



Figur 3.6: Plotting av *miile* mot kategori.

Ord	Ant. ord	bonde	embedsman	offiser	reindrift	sjøsame
Ant. ord	166 841	30 645	72 485	10 578	15 640	22 372
miil	2 168	435 (1,42%)	967 (1,33%)	28 (0,26%)	173 (1,11%)	373 (1,67%)
miile	365	155 (0,51%)	169 (0,23%)	0 (0,00%)	21 (0,13%)	1 (0,00%)
bøsseskud	305	3 (0,01%)	117 (0,16%)	10 (0,09%)	27 (0,17%)	95 (0,42%)

Tabell 3.4: Sum ord brukt til å angi lengder. % angir hvor stor andel av alle ordene til personer i kategorien det aktuelle ordet utgjør.

"bøsseskud" og andre lengdemål

Plottingen for "bøsseskud" vises i figur 3.4. "Bøsseskud" er en måte å måle avstand på. Det er da naturlig å sammenligne med de andre målene for avstand blant de 99 mest frekvente, *miil* og *miile*. Disse er plottet på figur 3.5 og 3.6.

Det ser ut til at entallsformen *miil* brukes relativt likt, mens flertallsformen *miile* brukes av bønder, til dels av reindriftssamer, og noe av sjøsamer. Derimot brukes ordet *bøsseskud* knapt av bønder, men mye av reindrifts- og sjøsamer.

Dette er nokså grove plottinger. For å se de reelle tallene, tok jeg ut alle kategorienes sum (menyvalg D9, jfr. tillegg B.5.7) og frekvens (menyvalg D8, jfr. tillegg B.5.6) og valgte ut de aktuelle ordene og kategoriene. Resultatet sees i tabell 3.4 og viser tydelig forskjellene. Entallsordet *miil* er i sum omlag tre ganger mer frekvent enn de to andre andre ordene til sammen. Det brukes tilsvarende av de tre kategoriene, andelen av alle ord i kategoriene varierer fra 1,11% til 1,67%.

Når det gjelder flertallsformen *miile*, ser vi at dette er ord som brukes en del av bønder (155 forekomster, 0,51%), mens det knapt brukes av sjøsamer (1 forekomst, 0,00%). Motsatt for *bøsseskud*, det brukes knapt av bønder (3 forekomster, 0,01%), mens det brukes en del av sjøsamer (95 forekomster, 0,42%). Reindriftssamer ligger i en mellomkategori, som tabellene viser.

I slike analyser er det viktig å se avvikende staveformer i sammenheng. For å sikre at det ikke bare var snakk om forskjeller i staving, tok jeg ut en KWIC-konkordans over alle ord som starter med *bøs* (menyvalg F3, jfr. tillegg B.7.3).[8]

I bøndenes avsnitt er i større grad *bøsseskud* stavet på avvikende måter, så som *bøsse-skud, bøssskud, bøsse skud* og lignende, enn hos de andre. Ved å korrigere manuelt for dette[9], kommer tabell 3.5 fram. Som man ser, er det ikke lenger så voldsomme utslag, men det er fortsatt tydelig at det er snakk om forskjellig bruk.

Når det gjelder *miile*, viser en gjennomgang av konkordansen at entalls- og flertallsformen brukes om hverandre. Det betyr at man kan slå sammen de to linjene, og forskjellene forsvinner.

Det er klart at personen som skriver ned en tekst, bestemmer stavemåten. Dette kan forklare at bønder, som er i flertall i andre protokoller enn sjøsamer,

[8] Jeg tok også ut en over alle ord som starter med *bør*, men der var det ingen former av "bøsseskud" for de aktuelle kategoriene.

[9] Systemet kunne her med fordel inneholdt en mulighet til å plukke ut enkeltord til analyse manuelt, men dette har jeg ikke implementert.

Ord	Ant. ord	bonde	reindrift	sjøsame
Ant. ord	166 841	30 645	15 640	22 372
bøsseskud	305	43 (0,14%)	43 (0,27%)	107 (0,48%)

Tabell 3.5: Sum for "bøsseskud", manuelt korrigert for avvikende stavemåter.

	Nodeid 5 964–53 028 (Røyem)	Nodeid 53 029–129 437 (Schnitler)
bøsseskud	2 (0,7%)	303 (99,3%)
miile	274 (73,1%)	101 (26,9%)

Tabell 3.6: "bøsseskud" og "miile" etter hånd.

har systematisk avvikende staveform. Når det gjelder bruk av to forskjellige lengdebegreper, er det mer sannsynlig at dette gjenspeiler ulik bruk hos vitnene.

Slik resultatene framkommer her, er det klart at det er visse forskjeller i ordvalg mellom de ulike gruppene. Dette kan skyldes hvordan forhørerne ønsket at vitnene skulle formulere seg, men det er sannsynlig at det er basert på vitnenes egne begrepsapparat. At bønder, som i større grad beveget seg langs oppmålte bygdeveier, brukte mil der samer i større grad brukte bøsseskud, er naturlig, selv om det klare utslaget for sjøsamer og reindriftssamer i en mellomposisjon er overraskende. Det antydes noe i den retning under et forhør i Namdalen:

> J almindelighed har Rætten mærcket af disse *Finne*-Viidner, at Siiden faae af dem har vanchet meget i bøjdene, hvor Miilene findes ved Stolper afpælede, Saa kan man ickke forsickre, at det Miil-Maal, Som de have opgiivet, Skal være riigtig. — (Schnitler 1962, s. 161)

Men det er en annen mulig forklaring som kan passe bedre med dette forholdet. Hvis vi nå husker tilbake til tabell 1.1 på side 15, minnes vi at teksten som finnes i nodene fra og med id 53 029 er skrevet av Schnitler, mens de tidligere er skrevet av Røyem. Hva med å sjekke "bøsseskud" og "miile" etter disse parametrene? Se tabell 3.6 for hva som da framkommer.

Merk at vitner klassifisert som sjøsamer utelukkende finnes i de nordlige delene av landet som er beskrevet i protokollene 5–6, mens bønder er sterkt overrepresentert i sør, protokoll 1–2. Merk også at denne siste undersøkelsen også inkluderer tekst som er tilordnet andre talere enn de tre kategoriene som først og fremst er vurdert. Tekstmengden totalt er således langt større.

For å se nærmere på eventuelt samsvar mellom hender og kategorier, kan vi se på hvilke avsnitt de ulike kategoriene er knyttet til (menyvalg B8, jfr. tillegg B.3.5). Ved å sette skille ved nodeid 53 029, kommer tabell 3.7 fram. Der ser vi at analyseresultatene som viser klar forskjell mellom bønder og sjøsamer med reindriftssamer som kategori mellom, samsvarer med skillet mellom de to skriverne av protokollene.

Vi har nå to ulike forklaringer på samme fenomen. Før jeg vurderer dem opp mot hverandre, vil jeg analysere flere ord.

	Node 5 964–53 028 (Røyem)	*Node 53 029–129 437 (Schnitler)*
bonde	386 (93%)	28 (7%)
embedsmann	529 (38%)	864 (62%)
offiser	4 (2%)	207 (98%)
reindriftssame	81 (26%)	225 (74%)
sjøsame	0 (0%)	424 (100%)
Sum	1058 (35%)	1963 (65%)

Tabell 3.7: Kategoriene og hvor mange avsnitt deres talere står for i de to ulike hendene.

"botten"

Dette ordet brukes i vårt materiale i forbindelse med nærmere angivelse av et sted, f.eks. bunnen av en fjord. Det er naturlig å tenke seg at "botten" er et ord som kan forekomme både i og utenfor stedsnavn. Det kan derfor tenkes at ulik taggepraksis, der man noen ganger tar med "botten" og andre ganger ikke, kunne være en årsak. For å sjekke dette, tok jeg ut en KWIC-konkordans der også innholdet i SNAVN-tagger ble analysert (menyvalg F2, jfr. tillegg B.7.2). Resultatet var det samme som viste seg i utskriften fra lisp-stat. Bønder bruker ordet 5 ganger (0,02%), reindriftssamer 29 ganger (0,19%) og sjøsamer 81 ganger (0,36%). Vi ser altså samme type fordeling som for forrige ord.

Jeg vil også her sjekke samme alternative forklaring som i forrige eksempel. I Røyems deler forekommer "botten" to ganger, mens i Schnitlers deler 279 ganger. Vi ser altså at samme forklaring er aktuell her.

"bredt" og andre med avvikende stavemåter

Ordet "bredt" er også stavet på andre måter i materialet. Jeg tok ut en konkordans på ord som begynte på "bre" og fikk da med andre aktuelle former. Det viste seg at "breed" gjerne brukes i samme forbindelse som "bredt". Hvis vi slår sammen disse to ordene, (tabell 3.8) ser vi at for summen av disse to er forskjellene ikke så påfallende.

Det samme forholdet finner vi for en rekke andre ordpar, se tabell 3.8. Jeg slo sammen "den" og "dend", "deres" og "deris", jeg koblet "een" til "en", slo sammen "han" og "hand", koblet "langt" til "lang", slo sammen "side" og "siide"[10]. Konkordanser bekreftet at bruken av ordene var tilsvarende for de ulike formene.

Felles for dem er at forskjellene ikke forsvinner, men at de ekstreme utslagene (ingen i en kategori, blant de 50 mest frekvente i en annen) erstattes av forskjeller opp til at den ene kategorien har dobbelt så mange forekomster som den andre.

"pag"

Denne forkortelsen betyr "side" og brukes når det henvises til tidligere forhør. I den trykte utgaven er sidetallene i manuskriptet skiftet ut med sidetallene i den

[10] Jeg regner da med at det samme gjelder for "siden" og "siiden".

Ord	*Sum ordformer*	*bonde*	*reindrift*	*sjøsame*
bredt	271	0,0000653	0,0015345	0,0032630
breed	334	0,0039811	0,0014066	0,0030842
Sum	605	0,0040464	0,0029411	0,0063472
den	1603	0,0035242	0,0119565	0,0124262
dend	868	0,0145538	0,0052430	0,0000000
Sum	2471	0,0180780	0,0171995	0,0124262
deres	288	0,0003589	0,0021100	0,0032630
deris	256	0,0016316	0,0005754	0,0000000
Sum	544	0,0019905	0,0026854	0,0032630
een	342	0,0056127	0,0004476	0,0002682
en	1254	0,0072769	0,0072890	0,0064366
Sum	1596	0,0128896	0,0077366	0,0067048
han	385	0,0006853	0,0081202	0,0058555
hand	761	0,0138032	0,0067775	0,0000000
Sum	1146	0,0144885	0,0148977	0,0058555
lang	538	0,0047642	0,0021739	0,0044699
langt	362	0,0008158	0,0024936	0,0032630
Sum	900	0,0055800	0,0046675	0,0077329
side	532	0,0001958	0,0042199	0,0059002
siide	327	0,0065916	0,0019182	0,0000000
Sum	859	0,0067874	0,0061381	0,0059002

Tabell 3.8: Bruksfrekvenser, dvs. antall ganger ordet er brukt i forhold til antall ordformer totalt, for “bredt” og “breed, “den” og “dend”, “deres” og “deris”, “een” og “en”, “han” og “hand”, “side” og “siide”.

Ord	*Sum ordformer*	*bonde*	*reindrift*	*sjøsame*
nord	868	0,0082232	0,0046036	0,0054979
norden	231	0,0016969	0,0021739	0,0015645
nordre	366	0,0024800	0,0019821	0,0030395
Sum	1465	0,0124001	0,0087596	0,0101019
østen	345	0,0023821	0,0021739	0,0026819
øster	736	0,0063306	0,0039003	0,0044699
østre	344	0,0001632	0,0026215	0,0034418
Sum	1425	0,0088759	0,0086957	0,0105936
søer	549	0,0034263	0,0044757	0,0067942
sønden	284	0,0003916	0,0028772	0,0033077
søndre	323	0,0002611	0,0027494	0,0046487
syd	66	0,0012074	0,0001279	0,0000447
Sum	1222	0,0052864	0,0102302	0,0147953
væster	254	0,0050253	0,0015345	0,0000000
vester	352	0,0000326	0,0012788	0,0038888
vestre	322	0,0001632	0,0018542	0,0042464
Sum	928	0,0052211	0,0046675	0,0081352

Tabell 3.9: Bruksfrekvenser, dvs. antall ganger ordet er brukt i forhold til antall ordformer totalt, for himmelretninger.

trykte teksten uten at dette er skrevet eksplisitt. "pag" brukes i større grad i vol. V enn i de andre volumene, men det må være skriverens valg, ikke talernes språkformer.

"vester", "sønden", "søndre", "vestre", "væster" og "østre"

Jeg sjekket alle ordformer som brukes om himmelretninger som forekommer blant de 99 mest frekvente ordene, med tillegg av ordet "syd". De finnes i tabell 3.9. På samme måte som ved sammenligninger over, viser det seg at forskjellene ikke er så store når man slår sammen ulike stavemåter. Men det er interessant å se hvordan sjøsamer bruker ulike ord for sørlig himmelretning nesten tre ganger oftere enn bønder.

3.4.4 Konklusjon frekvensanalyse

I dette analysearbeidet har jeg startet kvantitativt. Med utgangspunkt i resultatene fra kvantitative undersøkelser av større datamengder har jeg gått i kvalitativ retning på spesielt interessante data. Ved å ha vist at det finnes forskjeller i ordvalg, men at disse forskjellene kan forklares av substantivene i teksten, og ved å vise at det ikke kan påvises systematiske forskjeller for preposisjoner, er det sannsynliggjort at det ikke finnes systematiske stilforskjeller av tradisjonell type (Oakes 1998, s. 201–202) mellom det ordforrådet som ble nedtegnet basert på talerne i de tre aktuelle kategoriene.

I den grad jeg har funnet forskjeller, er det forskjeller som naturlig tilskrives

ulike temaer, ikke ulik stil. Når jeg spesifikt leter etter forskjeller som skulle kunne vise forskjeller i stil (ordlengde, funksjonsord), er det ikke mulig å påvise noen systematiske forskjeller.

Det har vist seg at det er et uheldig sammentreff i teksten ved at de delene som er skrevet av Røyem inneholder de aller fleste forhørene av norske bønder, og de delene som er skrevet av Schnitler inneholder alle forhørene av sjøsamer.

Vi har altså to konkurrerende forklaringer. Jeg vil ikke konkludere entydig med at kun den ene har betydning, men fordelingene hos reindriftssamene, hvis tale er nedtegnet av begge skrivere, og som tenderer til å havne mellom de to andre kategoriene, tyder på at skriveren har stor betydning. Dette motbeviser ikke eventuelle forskjeller mellom talerne. Men det øker usikkerheten ved resultatene og svekker argumentasjonen for at systematiske forskjeller mellom gruppene kan påvises.

3.5 Naboanalyse

Naboanalyse er en form for analyse der man ser på hvilke ord som tenderer til å opptre sammen med hvilke. I ytterste konsekvens lager man en $n * n$-matrise der n er antall ordformer i teksten. Dette tilsvarer det engelske begrepet "collocation" (Oakes 1998, s. 158 ff.).

Alle stedsnavn i teksten er tagget, jfr. avsnitt 1.5.2. I dette tilfellet vil jeg ta utgangspunkt i stedsnavnene, og jobber således med $1 * n$-matriser. Jeg ser på hvilke ord som forekommer bak stedsnavn, og hvilke som forekommer mellom to stedsnavn i et avsnitt. Hvordan stedsnavnene ble tagget er beskrevet i detalj i sluttrapporten fra delprosjektet i Dokumentasjonsprosjektet (Eide 1998 A, s. 43–44).

3.5.1 Ordet etter stedsnavn

I programsystemet er det verktøy som gjør det mulig å ta ut alle ord som forekommer rett etter et stedsnavn i samme avsnitt. Jeg har gjort en analyse som tilsvarer frekvensanalysen i forrige avsnitt, der ordene telles mekanisk og sammenlignes for personer i ulike kategorier. Det var ikke mulig å finne noen tendenser som samsvarer med kategoriene på samme måte som jeg fant for alle ord i avsnitt 3.4. Det er verdt å merke seg at antallet ord rett etter stedsnavn er nokså lite — det må nødvendigvis være mindre enn antall `snavn`-elementer, som er 13 883, jfr. tabell 3.3 på side 50.[11] Gjennomsnittlig antall ord pr. taler for våre tre kategorier er omlag 70. Når alle ordene ble analysert i avsnitt 3.4 var gjennomsnittet omlag 790 ord pr. taler.

3.5.2 Ordet mellom stedsnavn

Jeg vil nå avslutte med analyser som hovedsaklig består av manuell vurdering, hvor datasystemet først og fremst brukes til å sortere og gruppere materialet.

[11] Noen av navnene jo ikke noe ord rett etter seg i samme avsnitt.

Kontekst	*bønder*		*reindriftss.*		*sjøsamer*		*Max Δ*
	ord	*andel*	*ord*	*andel*	*ord*	*andel*	*(prosentpoeng)*
BLANK	61	15,10%	24	9,49%	20	6,47%	8,63
i	46	11,39%	34	13,44%	18	5,83%	7,61
eller	15	3,71%	10	3,95%	33	10,68%	6,97
fra	9	2,23%	7	2,77%	25	8,09%	5,86
ligger	13	3,22%	0	0,00%	4	1,29%	3,22
og	121	29,95%	73	28,85%	84	27,18%	2,77
denne	10	2,48%	0	0,00%	2	0,65%	2,48
er	10	2,48%	0	0,00%	7	2,27%	2,48
imellem	2	0,5%	2	0,79%	8	2,59%	2,09
dette	10	2,48%	1	0,4%	4	1,29%	2,08
hvor	2	0,5%	1	0,4%	7	2,27%	1,87
af	3	0,74%	0	0,00%	5	1,62%	1,62
imod	0	0,00%	0	0,00%	5	1,62%	1,62
ved	6	1,49%	5	1,98%	9	2,91%	1,43
hvilcken	5	1,24%	1	0,4%	0	0,00%	1,24
til	33	8,17%	18	7,11%	23	7,44%	1,05

Tabell 3.10: Ord mellom to stedsnavn som forekommer minst fem ganger i minst en av de tre kategoriene, med differansen mellom største og minste prosentsats.

Jeg vil se på hvilke ord som tenderer til å forekomme mellom to stedsnavn.

Jeg tok så ut en liste over alle kontekster som ikke er lengre enn ett ord, det vil si at de er blanke eller består av ett ord. Her får vi f.eks. med situasjonene der to stedsnavn står med en preposisjon mellom. Av disse trakk jeg ut de tre kategoriene jeg konsentrerer meg om og valgte alle ordformer som forekommer minst fem ganger i minst en av kategoriene[12], se tabell 3.10.

Jeg gikk så gjennom alle kontekstene der forskjellen mellom største og minste prosentandel i en kategori er større enn fem prosentpoeng. Til denne gjennomgangen brukte jeg en utskrift av kontekst etter `snavn` + ett eller flere ord, evt en `snavn` til for blank (menyvalg E7, jfr. tillegg B.6.5)

Kontekst: blank

Dette betyr at to `snavn` står rett etter hverandre. For å se hvilken sammenheng disse uttrykkene fantes i, slik at jeg kunne gruppere dem, tok jeg ut avsnittene de er en del av og så på den større konteksten de sto i. Det første som må kompenseres for, er en rekke dobbelttellinger. Når man har uttrykk som "SNAVN SNAVN SNAVN andre ord", blir dette talt to ganger. Det er flere slike lange uttrykk blant bønder enn de andre. Disse fjerner jeg fra tellingen for å se hvor mange uttrykk det totalt er snakk om. Etter å ha redusert for dem, kommer tabell 3.11 fram. Vi ser at forskjellen er redusert, det som var av forskjell mellom sjøsamer og reindriftssamer er nærmest borte, men bøndene skiller seg fortsatt klart ut.

[12]Færre enn fem ord er for lite til å analysere på grunnlag av (Oakes 1998, s. 201).

Kontekst	*bønder*		*reindriftss.*		*sjøsamer*		*Max Δ*
	ord	*andel*	*ord*	*andel*	*ord*	*andel*	*(prosentpoeng)*
BLANK justert	50	12,38%	18	7,11%	19	6,15%	6,23

Tabell 3.11: SNAVN SNAVN justert for flere tellinger av samme uttrykk..

Forhold mellom SNAVN'ene	*bønder*	*reindriftss.*	*sjøsamer*
Større — mindre	10 (20,0%)	3 (16,7%)	1 (5,3%)
Oppramsing	18 (36,0%)	10 (55,6%)	7 (36,8%)
Setningsskille, apposisjoner, flyttede preposisjoner	19 (38,0%)	4 (22,2%)	7 (36,8%)
Synonymer	1 (2,0%)	0 (0,0%)	0 (0,0%)
SNAVN i genitiv — SNAVN	2 (4,0%)	1 (5,6%)	4 (21,1%)

Tabell 3.12: Gruppering av kontekster "SNAVN SNAVN". Prosentene oppgitt er forholdet mellom SNAVN'enes andel av samlet antall kontekster for kategorien.

Jeg delte deretter inn setningene etter årsak til at flere stedsnavn kommer rett etter hverandre. Kategoriene og summene finnes i tabell 3.12. Her ser man at det er to viktige forskjeller. Den ene er at bønder i større grad bruker to stedsnavn som representerer "større — mindre", den andre er at bønder i mindre grad bruker "`snavn` i genitiv — `snavn`.

Kontekst: "i"

Her er det ikke aktuelt å gruppere bruken av uttrykket. Det brukes alltid til å vise at et sted er en del av et annet, og det er ikke mer å si foreløpig enn at den formuleringen brukes mindre av sjøsamer enn av talere tilhørende de andre to kategoriene.

Kontekst: "eller"

For å se hvilke typer stedsnavn som knyttes sammen av "eller" i hvilken kontekst, tok jeg ut en KWIC-konkordans med stedsnavnene skrevet ut (et vil si at de ikke er erstattet med "snavn") (menyvalg F2, jfr. tillegg B.7.2) og plukket ut "SNAVN eller SNAVN"-treffene. Jeg delte disse treffene i fire grupper vha. stedsnavnregisteret (Schnitler 1962, s. 438–473), se tabell 3.13. Skillet mellom kategoriene 2 og 3 er ikke lett å trekke, så forskjellene må brukes med forsiktighet.

Det vi ser tydelig, er at den store forskjellen dreier seg om ulike navn satt på samme sted (linje 2 og 3), og at det forskjellen mellom bønder og sjøsamer dreier seg om et begrenset antall steder — forskjellen i ulike uttrykk er bare 7 hos bønder mot 9 hos sjøsamer, mens forskjellen i samlet antall uttrykk er 10 mot 26.

Av de 26 formene hos sjøsamene er det 15 som utgjøres av to uttrykk: "Kiøfiord eller Neidensfjord" og "Bøgfiord eller Passvigfiord"[13]. Dette tyder på at det

[13]Ulike stavemåter av stedsnavnene og ulik rekkefølge er slått sammen.

Forhold mellom SNAVN'ene	*bønder*		*reindriftss.*		*sjøsamer*	
	totalt	*ulike*	*totalt*	*ulike*	*totalt*	*ulike*
Ulike stavemåter av samme navn	1	1	1	1	1	1
Ulike navn på samme sted, samme språk	10	7	1	1	21	5
Ulike navn på samme sted, ulike språk	0	0	6	3	5	4
Ulike steder	4	4	2	2	6	6

Tabell 3.13: Gruppering av kontekster "SNAVN eller SNAVN". Totalt er samlet antall forekomster, ulike er forekomster med ulike stedsnavn.

Forhold mellom SNAVN'ene	*bønder*	*reindriftss.*	*sjøsamer*
Hører ikke sammen (ulike setninger)	1	4	3
Lengdemål	5	2	15
Annet	3	1	7

Tabell 3.14: Gruppering av kontekster "SNAVN fra SNAVN". Lengdemål er uttrykk der det måles en kvantifisert avstand eller størrelse og begge stedsnavnene er involvert i samme måling.

er viktig å slå fast at et sted har flere navneformer i områdene der sjøsamer holder til, som i større grad er to- og trespråklige[14] enn områdene der norske bønder holder til, som i vårt materiale først og fremst er i bygdene lenger sør. Reindriftssamene forholdt seg også til en mer enhetlig språksituasjon enn sjøsamene — de måtte forholde seg til et nordiskspråklig stormsamfunn, men alle som utøvde samme næring som dem var samiskspråklige. Sjøsamer delte i større grad næringsvei med sine nordisk- og finskspråklige naboer.

Kontekst: "fra"

Kontekstene "SNAVN fra SNAVN" ble gruppert i tre kategorier i tabell 3.14. Klassifikasjonen var ikke like enkel for denne konteksten som for forrige, og kategoriene ble noe grove, med vanskelig definerbare grenser. Et eksempel på lengdemål er "*Skoggerøe*, fra *Kiøholmen* i Øster til Sønden 1/8 Miil [...]" (Schnitler 1962, s. 349), mens et eksempel på oppsett som ikke regnes som lengdemål er "*Reenøe* inde i *Passvig*-fiord, fra *Kielmøe* i Søer 1/2 Miil [...] (ibid.), fordi det ikke er de to stedsnavnene på hver side av "fra" avstanden måles mellom, men mellom "Reenøe" og "Kielmøe".

Som vi ser av dette forsøket på klassifikasjon, skyldes overrepresentasjonen av dette uttrykket hos sjøsamer først og fremst lengdemålene, selv om det også i "annet"-kategorien er flere uttrykk med sjøsamer som taler. Det er en lengdemålsformulering hos sjøsamene som ikke i særlig grad finnes hos de andre to kategoriene.

[14] I denne sammenheng regner jeg norsk, svensk og dansk som ett språk, nordisk, med ulike dialekter, og jeg slår også sammen samiske språk, slik at trespråklig innebærer norsk/svensk/dansk, samiske språk og finsk/kvensk.

3.5.3 Konklusjon naboanalyse

Jeg finner det vanskelig å trekke noen entydige konklusjoner ut av disse analysene, men vil først påpeke at det her ikke ser ut til å være noe system i at det er store forskjeller mellom bønder og sjøsamer med reindriftssamer som en mellomkategori. I utgangspunktet blir dermed ikke teorien om at skriverens hånd forklarer det meste av forskjellene bekreftet.

En annen mulig forklaring på de forskjellene vi finner er at deler av teksten som i de tidlige protokollene ble ført som oppsummering her i større grad ble ført som protokoller fra avhør av flere vitner. Jeg sjekket derfor "grunnene" (fra begrunnelsessystemet) til disse avsnittene, fordi de vil vise manuelt innskrevne kommentarer om gruppeavhør, som konsekvent ble skrevet inn når avsnitt ble koblet til taler.

Ved å ta ut en oversikt over alle begrunnelsene sortert etter talerkategorier på tilknyttede avsnitt (menyvalg G3, jfr. tillegg B.8.3) fant jeg ut at det ikke var noen bønder med gruppeavhør, mens det var 50 avsnitt hos reindriftssamer og 273 hos sjøsamer som representerer avhør av flere vitner. Ved å gå gjennom de avsnittene vi her har sett på og sjekket mot begrunnelsene (menyvalg G2, jfr. tillegg B.8.2), fant jeg at alle sjøsamenes avsnitt med unntak av ett representerer gruppeavhør, mens ingen av de andre gjør det.

Hvis vi går gjennom de ulike analysene og sammenfatter resultatene sortert på hvem som er sterkest overrepresentert, får vi følgende:[15]

Bønder overrepresentert

- `snavn snavn`: Større — mindre
- `snavn snavn`: Setningsskille o.l.
- `snavn` eller `snavn`: Ulike navn på samme sted, samme språk

Reindriftssamer overrepresentert

- `snavn snavn`: Oppramsing
- `snavn` i `snavn`
- `snavn` eller `snavn`: Ulike navn på samme sted, ulike språk

Sjøsamer overrepresentert

- `snavn snavn`: Første snavn i genitiv
- `snavn` eller `snavn`: Ulike steder
- `snavn` fra `snavn`: Lengdemål
- `snavn` fra `snavn`: Annet

[15] Grupper med svært få forekomster utelates.

En tydelig forskjell er at når man bruker "eller" mellom to stedsnavn, er de to stedsnavnene aldri på to ulike språk hos bøndene, mens de av og til er det hos de to samegruppene. Det er rimeligvis en konsekvens av at hovedspråket for bøndene er statens offisielle språk, mens samene i større grad bruker både norsk og samisk (evt. finsk) i navngiving.

Det er mindre bruk av "SNAVN i SNAVN" hos sjøsamer. Videre er det mindre tilsvarende bruk av "SNAVN SNAVN", der det er en inneholdt i-relasjon (større — mindre). I motsatt retning taler genitiv-konstruksjonene, men det er viktig å merke seg at det kun er snakk om fire forekomster.

Det ser altså ut til å være en viss forskjell i hvilke ord de tre gruppene bruker mellom stedsnavn, noe som kan ha sammenheng med forskjeller mellom geografisk uttrykksmåte, eventuelt også mellom andre typer formuleringer som tenderer til å framkomme i tolking fra samisk, men det er vanskelig å finne noe klart system i forskjellene. Dette har også sammenheng med at totalt antall forekomster ikke er så stort, så tilfeldigheter kan slå inn i større grad enn i analysene i avsnitt 3.4.

3.6 Oppsummering av analysene

3.6.1 Analysemetoder

De kvantitative metodene ble her først og fremst brukt til å finne fram til interessante data (lete-søke-metoden), ikke primært for statistisk å påvise forskjeller. Noen ting er lettere å finne i kvantifisert form enn f.eks. i konkordans-form. Tallkolonnene var nødvendige for å finne fram til ordene som skulle undersøkes.

Grupperingen av talerne ble delvis gjort før jeg gikk i gang med selve analysen. Det hadde kanskje gitt bedre resultater om jeg hadde ventet til analysemetodene var klarlagt og derved kunne tilpasset kategoriene og utvalgskriteriene bedre. På den annen side ville det kunne ført til en tilpasning som gav resultater ved å tilpasse spørsmålene til de resultater man ønsker. Da hovedtyngen av avsnitt var tildelt taler før analysearbeidet begynte, minsker risikoen for å preges av analysens konkrete mål. Og det viste seg jo at selv om jeg primært lette etter forskjeller mellom talere, dukket forskjellene mellom hender naturlig opp som en alternativ forklaringsmodell.

Jeg regner avsnittet som beskriver personen som skriverens oppsummering og tok det ikke med som avsnitt der personen er taler. Det kunne vært tatt med under personens egen stemme.

3.6.2 Konklusjon på analysene

Analysene viser et klart sammenfall mellom stil, ordvalg og hånd i teksten. Dette kan forklare alle forskjellene som ble funnet. Noen av forskjellene, som den mellom "bøsseskud" og "miile", kan også forklares som forskjeller i ordvalg mellom gruppene. Dette innebærer at tese 3 i avsnitt 1.7.1 ikke er bekreftet.

Forskjeller mellom gruppene kan ikke *påvises*. Som så ofte i språklig analyse, finnes det flere mulige forklaringer på de observerte fenomenene.

Skillet mellom de to skriverne samsvarte med skillet mellom norske bønder og sjøsamer. Det gjør det vanskelig å se hvilke forskjeller som skyldes skriver og hvilke som skyldes taler. De påviste forskjellene er heller ikke så store og interessante, så det er et spørsmål om man bør legge ned et større arbeid i å analysere teksten grundigere.

Man skulle i utgangspunktet forvente at bønder og reindriftssamer var mest forskjellige, med sjøsamer mellom, fordi sjøsamer i større grad delte næringsvei med norske bønder, og også levde tettere på det norske samfunnet. Det innebærer at de resultatene som er funnet her ikke tyder på at det er noen skala av "samiskhet" i tekstene. Det kan være et argument for at hendene er hovedforklaringen på forskjellene, men det kan også peke i retning av at en slik "samiskhet" ikke ble uttrykt i forhørene. Det behøver ikke betyr at det ikke var systematiske forskjeller mellom norsk og samisk geografisk tenkning, men det kan hende at samene kjente til den norske måten å snakke om geografi på og tilpasset seg denne i forhørssituasjonen — i avsnittet om tingbøker (avsnitt 1.6.3) ble det jo hevdet at forskere kjenner eksempler på at vitner sannsynligvis snakket forhøreren etter munnen. Dessuten kan tolken i oversettelsen ha tilpasset vitnemålene til samme geografiske system som norske bønder brukte.

Dersom det vi har funnet er forskjeller mellom Røyem og Schnitler, og disse er stilforskjeller, skulle man forvente at disse også viste seg i figur 3.2 på side 52. Det betyr også at dersom ulik hånd forklarer forskjellene mellom (særlig) sjøsamer og bønder, så er ikke hendene forskjellige på funksjonsord, noe som tyder på at de er relativt stillike.

Når det i avsnitt 1.7.1 skrives at påstanden om disse tekstene som kilder til geografiske tankesett ikke kan avvises på grunnlag av denne undersøkelsen selv om undersøkelsene gir negativ resultat, må dette modifiseres noe. Det er et argument mot forskjellene at det ser ut til å være et sterkt samsvar mellom hånd og stil i teksten, fordi dette er forskjeller som ser ut til å være så sterke i forhold til de individuelle forskjellene at de slår gjennom i analysene. På den annen side: Jeg kan ikke se bort fra muligheten for at metodene brukt i mine analyser ikke er gode nok, at det finnes tydelige forskjeller jeg ikke klarte å avdekke. Bare videre forskning kan finne ut av det.

3.6.3 Videre analysearbeid

Selv om det er klare indikasjoner på sammenhengen mellom hånd og stil i teksten, hadde det vært interessant å sjekke dette ved å gjøre en ny analyse med samme metoder, der avsnittene, evt. avsnitt tilordnet taler, ble gruppert etter hvem som skrev ned teksten, uavhengig av talerens stand. I den forbindelse hadde det også vært interessant å studere hele formuleringsprosessen, fra ordene sies som respons på et spørsmål, via presiseringer og evt. tolking til nedtegning og evt. rettelser i teksten. Ulike formuleringer kan også skyldes ulike spørsmål, f.eks. har vi eksempler på at en same stilles delvis andre spørsmål enn de foregående norske bøndene (Schnitler 1962, s. 150). Men for den slags analyser er

datagrunnlaget fattig, selv om slike ting som hvilke personer som tolkes for sin tid er godt dokumentert.

Det hadde vært nyttig å legge inn alder på alle personnavn med kobling til `pnavn`-noder. Men det må i tilfelle gjøres manuelt, da opplysningene ikke er registrert i registeret, selv om det i de fleste tilfelle står i teksten under innledningen til hvert forhør. Da kunne man gjøre tilsvarende analyser som i denne oppgaven og lete etter forskjeller som følger alder.

Det er åpenbart at valg av kategorier teksten ble delt inn i var avgjørende for analysen. Det kunne vært gjort annerledes. "Uspesifisert skriver" kunne vært delt i en rekke ulike kategorier etter tekstenes funksjon fra det spesifikke til det mer og mer sammenfattede og generelle, så som reisebeskrivelse, dagbok, forslag til organisering, lovutkast, osv. Burde vi i stedet for bare personkategori sett på funksjon, og delt teksten i *beskrivelse av reise, edsavleggelser o.l.* (embedsmann), *forhør/spørsmål* (embedsmann) og *svar* (arbeider, bonde, reindriftssame, sjøsame)?

En metode jeg ikke hadde ressurser til, da den ville krevd grammatisk oppmerking av teksten, er å se etter setningsmønstre. Til hver node som representerer et avsnitt som skal analyseres knyttes det to datastrukturer. En er selve den grammatikalske taggingen, representert med datastrukturen `(pcdataid offset lengde grammatikk)`. Den andre er det sett av mønstre som finnes i avsnittet. Et mønster kan f.eks. være (konj adv pron verb adj adj subst konj adv verb verb adj subst prep). Dette gjør det mulig å se hvor mange ganger pr. avsnitt et bestemt delmønster, f.eks. (verb adj+ subst), går igjen.

De konklusjonene som her har framkommet er ikke nødvendigvis noe argument mot tingbøker som kildemateriale, da protokollene analysene i denne oppgaven er basert på er ført med et bestemt formål, og ikke av sorenskrivere. Det er klart at et materiale med større grad av veksling mellom talere av ulike kategorier hadde vært bedre for en analyse som dette. Da kunne man sammenligne personers uttalelser som ble skrevet ned av samme person på omtrent samme tid. Tingbøker kunne vært et alternativ, men de er ikke i samme grad som Schnitler konsentrert rundt geografi, som var av særlig interesse i denne studien.

Det er uansett nyttig å huske at visse euforiske utsagn om hvor godt slikt materiale er som kilder, jfr. avsnitt 1.6.3, virker underlige i lys av denne undersøkelsen. Det kunne absolutt vært interessant å bruke lignende metoder på tradisjonelle tingbøker for å undersøke om det der ser ut til å overleve systematiske stilforskjeller fra vitner gjennom nedtegningen inn i håndskriftet.

Kapittel 4

Oppsummering

I de to foregående kapitlene har jeg beskrevet programsystemet som er utviklet i denne oppgaven fra to ulike sider. Kapittel 2 tok opp oppbyggingen av selve systemet, mens kapittel 3 viste hvordan systemet ble brukt i analyse av en tekst. I dette kapitlet vil jeg ta opp igjen tråden fra kapittel 1 og diskutere det systemet jeg har laget og brukt ut fra den bakgrunnen jeg der skisserte opp. Hvordan forholder mitt system seg til utviklingen av datasystemer for tekstanalyse? Hva kan det bety for den videre utviklingen av digitale kildehenvisninger? Hvilke muligheter åpner seg videre?

4.1 Verktøyet

Et verktøy er et arbeidsredskap, oftest spesialisert for en jobb. I denne oppgaven står to ulike verktøy sentralt:

1. Det verktøyet jeg brukte til å lage programsystemet: Common Lisp (GNU CLISP 2001).

2. Det verktøyet jeg laget i og med utviklingen av programsystemet, og som ble brukt i analysen i kapittel 3

4.1.1 Common Lisp

Det finnes mange verktøy som kunne vært brukt til å utvikle programsystemet i denne oppgaven. Valg av verktøy bør ikke være avgjørende for de resultatene man kommer fram til så sant metodene man bruker er sunt implementert.[1] Da jeg startet arbeidet hadde jeg valget mellom to aktuelle muligheter: Et dataprogram med tekstbitene lagret i strukturer i minnet og en databasebasert løsning. I mitt tilfelle var et objektorientert dataprogram med lagring i minnet best. Jeg skulle utvikle et enbrukersystem som skulle fungere som en prototype, og trengte

[1] Valg av verktøy vil alltid påvirke resultatet av en arbeidsprosess i noen grad, men jeg vil ikke diskutere det i denne oppgaven.

ikke det som er databasesystemenes viktigste fordeler, flerbrukerhåndtering og effektiv behandling av store datamengder. Programsystemet er laget som en testbenk, der det ikke stilles samme krav til ytelse som i et produksjonssystem, og der datamengden som representeres ikke er veldig stor.

Valg av programmeringsspråk var basert på pragmatiske kriterier. Lisp ble valgt fordi det er et språk som egner seg for raskt å utvikle kjørende prototyper, såkalt "rapid prototyping" (Graham 1996, s. 3, jfr s. 401). At dette fører med seg lavere kjørehastighet har ikke vært noe stort problem i denne oppgaven. Jeg har valgt ut enkelte funksjoner som jeg har bearbeidet for å øke hastigheten, mens jeg har latt andre forbli langsomme fordi det ikke skaper problemer i bruken av systemet. De prosessene som har tatt lang tid i kjøring har vært rutiner uten interaktivitet, så som import av SGML-filer, oppbygging av analysestrukturer og lignende. I de interaktive rutinene holder systemet tritt med brukerens arbeidshastighet.

Det er også en fordel at programmeringsspråket har et integrert grensesnitt der man kan kjøre Lisp kombinert med de funksjonene jeg har implementert som et interaktivt system. Dette innebærer at brukeren i tillegg til å kunne kjøre mine forhåndsdefinerte rutiner fra menysystemet, kan kjøre funksjonene direkte fra Lisp-promptet. Mer om dette nedenfor.

4.1.2 Informasjonssystemet

Det verktøyet jeg har utviklet gjør jobben det er laget for, jfr. kapittel 3. Data blir lastet inn i systemet, de blir strukturert i samarbeid med brukeren, og analysedata blir gjort tilgjengelig, dels for menneskelig vurdering, dels for eksterne rutiner. Det er likevel svakheter i programsystemet. Noen av dem har jeg kommet inn på i de foregående kapitlene, noen flere vil nevnes her.

I en del tilfelle har det vært nødvendig å endre rutiner for å få opp hastigheten. Dette fører gjerne til at elegante løsninger (f.eks. å traversere data ved bruk av strukturen i det DOM-lignende treet) må erstattes med mindre elegante metoder, f.eks. å løpe gjennom id-numre og aksessere objektene via hashtabeller.

Det er viktig å skille mellom to ulike slags svakheter. Den ene, og alvorligste, er når systemet gjør noe, men ikke gjør det korrekt. Den andre er mangel på funksjonalitet. Svakheter av den første typen har jeg fjernet så langt jeg har funnet dem, med unntak av falske positive i koblinger av data diskutert i avsnitt 2.3.3. Når man forsøker å gjøre noe med svakheter av den andre typen, kan man imidlertid risikere at løsningen i seg selv blir et problem. Enkelte applikasjoner utvikler seg til systemer som forsøker å gjøre alt. Det fører gjerne til at de blir uhåndterlige, både i bruk, vedlikehold og videreutvikling.

Men dersom man ikke forsøker å løse et bredt spekter av problemer i systemene man utvikler, blir oppgavene hvert system kan brukes til så få at man gjerne må utvikle nye systemer for hver ny oppgave. Det kan føre til at utviklingskostnadene blir uoverkommelige. Jeg mener derfor at modularisering, med mulighet for nettverksbasert sammenkobling av moduler utviklet i ulike miljøer, er den eneste veien å gå i utvikling av applikasjoner for humanistisk tekstanalyse. Jeg vil komme tilbake til dette i avsnitt 4.2.1, men vil først ta opp en enklere form

for modularisering jeg har implementert i programsystemet i denne oppgaven, nemlig eksportrutiner.

Eksport av data

Det er nødvendig for et godt system å kunne eksportere data. Dette gjelder ikke minst i de tilfellene der brukeren ønsker å bruke andre applikasjoner til videre arbeid med data fra systemet. Jeg har laget slike filtre for eksport til flere ulike formater: Tekst i kolonneformat med skilletegn til SPSS, kjørbar Lisp-kode til Lisp-stat, HTML til framvisning i vevleser.

Dette er en innarbeidet metode i humanistisk forskning. Slik datautveksling gjøres enten på applikasjon til applikasjonsbasis, slik jeg har gjort med skreddersydde eksportfiltre, eller man kan eksportere til og importere fra et standardformat for utveksling av data. Eksempler på slike standardformater er TEI (TEI 2002) og CIDOK-CRM (CIDOK CRM 2004). Slik eksport-import er diskutert i avsnitt 4.2.3.[2]

Et viktig problem med eksport-import er at man bryter lenken fra kilde via data til analyse. Man får duplisering av data (Thorvaldsen 1999, s. 196). Det betyr at endringer som gjøres i de data som ligger i systemet man eksporterer fra ikke reflekteres i analysen uten ny eksport-import, og at endringer som gjøres i systemet data er importert til etter import ikke reflekteres tilbake på data i systemet de ble eksportert fra. Dette er i og for seg greit slik jeg har gjort analysene, fordi jeg ikke har gjort noen videre bearbeiding av mine data i analyseverktøyene. Dersom man trenger videre datamassasje i analyseapplikasjonen, særlig når denne massasjen innebærer valg som gjøres av en menneskelig operatør, skaper det imidlertid problemer. Det er her nettverksbaserte løsninger kommer inn, jfr. avsnitt 4.2.1.

Interaksjon med brukeren

De mest aktuelle rutinene for en bruker av systemet er pakket inn i menysystemet, men alle funksjonene kan også kalles fra Lisp-promptet. Noen av funksjonene er definert med standardparametre i menysystemet, men de kan kalles fra Lisp-promptet med andre parameterverdier. Selv om muligheten til å kalle rutinene direkte er spesielt viktig under utvikling og for prototype-systemer, kan det være nyttig også i produksjonssystemer at brukeren kan kalle funksjoner på denne måten. Det gjør skripting og utvikling av skreddersydde rutiner enkelt for brukere med den nødvendige kompetanse. Men hvordan er det å bruke et slikt program hvis man ikke selv har mulighet til å skrive tillegg ved behov, eller ikke forstår hvordan slike tillegg er skrevet? Bør den humanistiske forskeren også være programmerer? (Thorvaldsen 1999, s. 11).

En bedre løsning for kommunikasjon med brukeren kunne være å bruke et metaspråk for å spesifisere uttak i stedet for å implementere alle analyseuttak direkte i Lisp, jfr. neste avsnitt.

[2]Som omtalt i avsnitt 2.2 har også systemet mulighet for å lese fra og lagre til SGML basert på databeskrivelsen de digitale tekstene opprinnelig ble tagget i.

Mangel på parametrisering

Flere steder i programsystemet er det utviklet rutiner som er svært like, f.eks. de som eksporterer data til ulike applikasjoner og de som gjør henholdsvis frekvensanslyse og naboanalyse. Det ville redusert kodemengden og forenklet vedlikehold og videreutvikling dersom slike rutiner hadde blitt skrevet sammen og ulikhetene ble styrt av parametre.

Det hadde også vært nyttig med et enklere system for vedlikehold og oppdatering av navneobjekter. Spesialiserte navn er nå lagret hardkodet i stedet for å ligge i oppdaterbare variable. Dette er uttrykk for et mer generelt problem som har sammenheng med at utvikler og bruker har vært en og samme person: For mye av det faglige innholdet i programsystemet er hardkodet i funksjonene i stedet for å leses av funksjonene som spesifikasjoner som ville vært lettere å oppdatere for brukeren. Med andre ord: Systemet er i for stor grad basert på analyse i koden, f.eks. i funksjoner som `plukk-sider` (kode på side 114) og `slaa-sammen-med-parantes` (kode på side 107). Kriterier burde vært synliggjort ved at de lå utenfor selve algoritmene.

Menysystemet gjør dette på en bedre måte. Det består av en funksjon som leser ei spesifikasjonsfil (kode på side 131). Dette gjør det f.eks. enkelt å oversette denne delen av systemet til engelsk. En slik deling mellom funksjoner som gjør en generell oppgave og parameterfiler som spesifiserer detaljene burde vært brukt flere steder i systemet.

4.2 Veien videre

4.2.1 Hvilket system bør lages?

Dersom man skulle lage et produksjonssystem basert på programsystemet i denne oppgaven, bør hele systemet reimplementeres. LISP-programmet kan i tilfelle fungere som en spesifikasjon. Om man gjorde en slik reimplementasjon, ville resultatet blitt et produksjonssystem som ligner sterkt på programsystemet i denne oppgaven.

Imidlertid bør nok et produksjonssystem være et flerbrukersystem og vil ha behov for dataintegritet og flebrukeradministrasjon som gjør en databasebasert løsning mer egnet. Å lagre SGML-dokumenter i databaser er en vel utprøvd metode. Det kan gjøres i en vanlig relasjonsdatabase der hver node har en linje i databasetabellen, eller i spesialiserte databasesystemer som er laget for å behandle SGML og XML-dokumenter. F.eks. kan man i Oracle bruke den støtten som finnes for DOM (Oracle 2002, kap. 2), og skrive de funksjonene man trenger i tillegg, enten som kode i basen eller i eksterne applikasjoner koblet til basen.

En interessant løsning i så måte ville vært å lage rutinene basert på Text Encoding Initiative-standarden (TEI 2002), slik at de ville være standardiserte. For materiale som det jeg jobber med, som ikke er kodet i TEI, måtte man da lage konverteringsfiltre, enten i systemet eller utenfor.[3]

[3]Se avsnitt 4.2.3 om hvordan systemet laget i denne oppgaven kan bygges ut med mulighet

Men det er ikke slik at det eneste som skiller programsystemet i denne oppgaven fra et godt produksjonssystem er ting som kan løses i en ren reimplementasjon. Det er viktige områder som mangler i dette systemet som bør være med i et produksjonssystem. Et eksempel på dette er muligheter for grafisk framvisning og manipulasjon av objektene, både de i det DOM-lignende treet og de andre. Et annet er statistisk analyse og visualisering.

Det er klart at det hadde vært mulig å gjøre alle analysene i Lisp-stat og integrere dette i programsystemet, som jo også er implementert i Lisp og som derfor kan koble til seg funksjoner fra Lisp-stat. Det hadde således latt seg gjøre å inkludere muligheter for å gjøre for eksempel PCA, som i denne oppgaven er gjort eksternt i SPSS, i systemet. Men med en slik metodikk går man glipp av flere muligheter som burde finnes i et godt produksjonssystem. Eksempler på dette er:

1. Det er ikke enkelt for andre å analysere mitt materiale videre i andre applikasjoner.

2. Det er vanskelig å bruke mitt system til å analysere andre slags tekster, som tingbøker fra ulike land.

Dessuten kan man ikke implementere *alle* tenkelige metoder i ett og samme system. Løsningen er ikke å lage et stort system som inkluderer alle muligheter. Løsningen må ligge i en form for modularisering, gjerne koblet til en database-basert implementasjon der basisfunksjoner kobles til det publiserte materialet.

Dette er metoder foreslått av flere. John Bradley foreslår en nettverksbasert modell der DOM-trær som representerer TEI-kodede dokumenter kan kobles til analyseverktøy på andre maskiner.

> This direct access can be provided when the DOM model is combined with technologies such as CORBA and RMI [...] a DOM-based document could be stored on a remote computer and a user would see it just as if it were in fact stored on his/her local computer — although the access would seem slower. (Bradley 2000, avsn. 4.2)

I et slikt system kan data i DOM-treet, applikasjoner og brukergrensesnitt ligge på samme maskin. Men man kan også organisere det annerledes, f.eks. med applikasjonen og brukergrensesnittet på én maskin og DOM-treet på en annen maskin man har tilgang til over nettet. DOM-treet og applikasjonen kan også ligge på én maskin mens brukergrensesnittet ligger på en annen, eller man kan dele det på tre ulike maskiner (ibid, avsn. 4.3). Ulike deler av DOM-treet kan også ligge på ulike maskiner (ibid, avsn. 4.4). Lignende systemer er også foreslått av Stéfan Sinclar (Sinclar 2004) og Geoffrey Rockwell (Rockwell 2003).

Man bør arbeide i samme retning hva utvikling av analyseverktøy angår som man er i gang med for kildemateriale, jfr. neste avsnitt: Man kobler sammen fra ulike kilder. Man må gjøre det mulig å integrere rutiner fra ulik programvare

for eksport til TEI.

og å koble ulike systemer og tekster sammen, for analyse, for utforskning, og kanskje for å skape ny kunst? (ibid)

Et av problemene med en slik løsning i dag er utbygging av nettkapasitet, både hastighet, dekningsgrad og pris. Selv om nettbaserte verktøy er en god ide, kan det være nødvendig at noen interaktive operasjoner skjer lokalt, fordi nettet skaper stadige forsinkelser i arbeidet. Det er også et spørsmål om i hvilken grad man skal gjøre seg avhengig av et tilgjengelig og fungerende datanett.

Men det er også et mer generelt problem med ideer som dette: De er lettere å se for seg på tegnebordet enn i virkeligheten. Man har sett mange gode ideer til systemer for humanistisk forskning, som Manfred Thallers "Historical Workstation". Det er et godt system, hvis eneste mangel er at det kun finnes som idé (Thaller 1991). Det betyr ikke at slike ideer ikke er viktige, det er nyttig å ha tidligere tiders vyer med seg når brikkene som er nødvendige for å bygge integrerte systemer langsomt faller på plass, men man må skille klart mellom hva som er vyer og hva som i praksis er implementerbart i dag. Eksempler som TAPoR (Rockwell 2003) og HyperPo (Sinclair 2003) er derfor viktige, fordi de viser noe som faktisk er implementert.

4.2.2 Kobling til andre kildesamlinger

En tekst, også en gammel tekst, er forståelig når man kan språket den er skrevet på. Men for mange lesere oppstår det problemer med de pragmatiske aspektene i gamle tekster. Mange av referansene kan være uforståelige for en moderne leser.

Konteksten til Schnitlers protokoller gjorde dem forståelige for samtidige lesere, og de er fortsatt godt forståelige for en moderne leser med et visst kjennskap til 1700-tallets Norden. Men mange opplysninger i teksten, som stedsnavn og personnavn, er ukjente for de fleste moderne lesere, også historikere. Eksempler på slike referanser som er vanskelige å sette inn i sammenheng for en moderne leser, særlig når man får seg forelagt korte fragmenter, er det flere av her:

> **Ao 1744. d. l7 Sept:** satte man Retten paa *Fladtøe*, nærværendes Lensmand *Simon Simonsen Maasøe*, med LaugRettesMænd, over de fra *Refsbotten* ankomne *Finne*-Vidner; For hvilke ved Tolken *Hælset* Eeden af Lovbogen blev forklared; Som og derpaa aflagde deres *Corporlig* Eed: Efter vedtagen Orden erkyndigde man sig forud hos Lensmand og LaugRetten, samt hos Vidnerne, om Landets Leje og Strekning af dette Sted og Præstegield, fra SøeKanten op i Søer til Grændsen: (Schnitler 1962, s. 257)

I sammenhenger der man skal kommunisere med personer med bakgrunn fra mange ulike kulturer, forsøker man i størst mulig grad å være kontekstfri, som man f.eks. ser på skilt på flyplasser. I motsatt ende av skalaen finner man f.eks. spørsmål man kan lagre for å hjelpe en til å huske et passord, hvor målet er at mangler på kontekst skal gjøre det umulig for alle andre å finne svaret. Dette minner om Daniel L. Smails skille mellom geografiske identifikatorer som "mnemonic devices for notaries" og som "formal, impersonal markers of identity" (Smail 1999, s. 202). Spørsmålet er hvor vid kontekst som kreves for å

forstå en tekst. Etterhvert som tiden går etter at en tekst er skrevet ned, vil standardkonteksten til leserne forskyves, slik at det stadig blir vanskeligere og mer arbeidskrevende å finne mening.

I tillegg til dette, flytter man tekster ut av sin tradisjonelle kontekst gjennom trykking og digitalisering, og gjennom oppstykking fjerner man ofte den nære konteksten når man flytter tekster over i analyseverktøy. Da får man behov for rekontekstualisering for at en leser skal kunne forstå tekstfragmentene man sitter igjen med En måte man kan rekontekstualisere tekster på, er å koble meningsbiter i dem til andre tekster.

I denne oppgaven har jeg koblet data fra personnavnregisteret tettere til data i hovedteksten i Schnitler-utgaven, og i tillegg lagt til ny informasjon. I forbindelse med digitalisering av kildemateriale i Dokumentasjonsprosjektet skrev jeg om kobling mellom data fra ulike kilder, særlig tekster (Eide 1998 B). Her vil jeg gå gjennom noen koblinger som jeg ikke har implementert, men som peker seg ut som gode kandidater for videre arbeid i retning av et hypertekstsystem basert på Schnitler og andre kilder.

Geografiske koblinger

Det er gode muligheter til å koble stedsnavnene som nevnes i Schnitler til andre tekster og til kart. Dette skyldes at stedsnavnregisteret har kartreferanser til alle stedsnavn som lot seg identifisere, som i dette eksempelet:

> *Aardejok* 337 = Ordojokka eller Hjordelva, lenger aust Vesterelva (= Syltevigfiord-Elv), midt på Æ 3 Båtsfjord. (Schnitler 1962, s. 438)

Æ 3 Båtsfjord er en kartreferanse som er forklart i innledningen til registeret, den henviser til et blad i *Topografisk kart over Norge* i målestokken 1:100 000 (ibid, s. 437)

Man kan lage stedsnavnobjekter som tilsvarer de personnavnobjektene som har blitt brukt i denne oppgaven. Følgende trinn er da aktuelle å gjennomføre:

1. Lag stedsnavnobjekter til alle navn som er oppgitt i stedsnavnregisteret.

2. Legg inn en kobling til hvert stedsnavn i teksten basert på sidehenvisningene i registeret og navnelikhet.

3. Koble stedsnavnobjektene til geografiske referanser i form av flater definert ved UTM-koordinater. Dette gjøres ved å bruke kartreferansene i stedsnavnregisteret, f.eks. ved å legge inn koordinatene for hvert kartblad i en utvekslingstabell.

Dette er alt som skal til for en grov kobling mellom alle identifiserte stedsnavn i teksten, andre tekster og digitale kart. Det som mangler er en mer nøyaktig plassering av hvert enkelt sted. Det er det også mulig å få til for mange av stedsnavnene, men dette krever mer manuelt arbeid.

Personkoblinger

Når det gjelder personnavnene, ble mye av jobben som er skissert for stedsnavnene ovenfor gjort i arbeidet med denne oppgaven. Men det å koble mot eksterne kilder er vanskeligere, fordi man ikke har noen klar definisjon av en historisk person på samme måte som geografiske koordinater fungerer for steder. Det betyr at kobling til eksterne kilder (kirkebøker, tingbøker, misjonsdokumenter, osv.) krever betydelig manuelt arbeid. Her er det åpenbart at et nasjonalt historisk personnavnregister, som er foreslått av flere, hadde vært et nyttig verktøy, helst med koblinger til tilsvarende registre i andre nordiske land.

4.2.3 Eksport: TEI og CIDOK-CRM

Tekstene som ble lest inn i programsystemet fra SGML-filer kan skrives ut igjen i SGML. En videreutvikling ville være å lage et filter til TEI i programsystemet, slik at teksten kan eksporteres i TEI-format. Det ville i tilfelle vært en syntaktisk transformasjon der det meste kunne gjøres ved å endre navn på nodeelementer i det DOM-lignende treet. Dette ville neppe i seg selv være en mer effektiv metode for konvertering til TEI enn verktøy som allerede finnes, f.eks. basert på XSLT (XSLT 2003), men utviklingskostnadene når man først har programsystemet slik det nå står er ikke så store.

En tilleggsgevinst ville være at en eksport til TEI kunne inkludere data lagt til i analysen. TEI inneholder imidlertid ikke noen veldefinert måte å spesifisere mange av datatypene som er laget i denne oppgaven på, og jeg mener at det ikke er noen naturlig oppgave for TEI å modellere slike data. Det man derimot kan gjøre er å eksportere data som ikke passer i TEI, f.eks. detaljerte persondata og analysedata, til en standard som passer bedre for slike data, som CIDOKs Conceptual Reference Model (CIDOK CRM 2004). Om dette legges i samme dokument som TEI-teksten eller i et separat dokument er kun et teknisk spørsmål, TEI har uansett definert måter å koble inn ikke-TEI-materiale på (TEI 2002, kap 15-16). Dette vil da innebære at grunnteksten eksporteres i TEI-format, mens de hendelsene som ble spesifisert og lagret under arbeidet i denne oppgaven legges i CIDOK-CRM-delen.

Så kunne man bruke id-koblinger til å koble TEI-dokumentet og de CIDOK-CRM-definerte dataene sammen. Slike pekere vil kunne brukes begge veier, fra analyse til kildetekst og fra kildetekst til påstander om inneholdet i teksten og analyser av den. Dette tilsvarer måten datastrukturene ligger lagret på i programsystemet, som separate datastrukturer som er koblet sammen med pekere. Arbeid for å integrere TEI med ontologier som CIDOK-CRM på denne måten er i gang i en Special Interest Group under TEI Consortium som ble opprettet under TEIs 4. årlige møte i Baltimore 22.-23. oktober 2004.

Dette betyr at etter en slik eksport har en tolkning av utvalgte deler av innholdet i den opprinnelige SGML-teksten blitt mappet inn i CIDOK-CRM. En interessant effekt av dette er at eventuelle problemer i mappingen til CIDOK-CRM kan tyde på problemer med kvaliteten på taggingen og det etterfølgende arbeidet i denne oppgaven, dvs. kvaliteten på datamodellen som er brukt

(Jordal 2004).

Et av problemene som vil oppstå i en slik eksport skyldes en svakhet i modelleringen av begrunnelsessystemet: Tidspunktet for hver hendelse ble ikke lagret, slik at rekkefølgen på hendelsene ikke er eksakt registrert.

Det ville også vært interessant å se hvordan man kunne implementert alternative påstander om hvordan teksten skal forstås, noe som er mulig i en hendelsesorientert modell som CIDOK-CRM. Dette ville ikke minst være viktig dersom man skal utvikle flerbrukersystemer for analyse.

4.3 Begrunnelser eller kildehenvisninger?

4.3.1 Delvis formalisering

Mekanisk spesifiserte hendelser er lette å formalisere. Gunnar Thorvaldsen mener det er et argument for mekaniske metoder med klart definert kodebok at en slik metodikk gjør det lettere å etterprøve forfatterens arbeid (Thorvaldsen 1999, s. 109, jfr. ss. 139, 141, 142–143). Men det er ikke alle avgjørelser som kan implementeres i mekaniske regler uten at systemet blir urimelig komplekst. Man må akseptere at brukeren interagerer med systemet og gjør valg som påvirker dataene, men man bør søke å dokumentere slike valg best mulig. I tillegg til hva som ble valgt, er det ting brukeren vet i det han gjør sine valg som det ikke er mulig å formalisere, i hvert fall ikke der og da, men som likevel er nyttig å ta vare på.

Begrunnelsessystemet gjør det mulig å legge inn informasjon i systemet som jeg ikke har laget formaliserte rutiner for å lagre. Det er altså mulig å gjøre informasjon eksplisitt, selv om den ikke kan formaliseres. Det er snakk om en form for annotering som minner om det som er kjent fra CAQDAS-systemer, jfr avsnitt 1.3.3. Begrunnelsene fungerer som et tillegg til selve valget, slik at en kan spesifisere årsaker knyttet til valg.

Det betyr at vi får et sett av formaliserte hendelser der den menneskelige inngripen dels lagres i sin konsekvens, og dels har mulighet til en lagret begrunnelse, kommentar eller annotering skrevet inn av brukeren i valgsituasjonen. Implisitt eller taus kunnskap kan på den måten gjøres eksplisitt.

Begrunnelsessystemet gir oss altså et informasjonssystem som utvider lagringen av to typer valg med en tredje informasjonsgruppe. De to valgtypene er:

1. Manuelle valg: Automatisk registrering av valgets konklusjon.
2. Mekaniske valg: Automatisk registrering av valgets prosedyre og konklusjon.

Det vi får i tillegg er:

3. Manuell registrering av begrunnelser knyttet til manuelle valg. Her kan f.eks. ting som man etter hvert kan utvikle en kategorisering av havne før analysen har vist behovet og muligheten for en slik kategorisering.

Mange valg lagres i systemet uten mulighet for kommentarer fra brukeren, fordi de er gjort uten brukermedvirkning. Men for valg med brukermedvirkning gis altså brukeren mulighet til å uttrykke bakgrunnen for de valgene som gjøres. I neste omgang kan slike begrunnelser fungere som forslag til systematiske, formaliserte klasser. Dette minner om CAQDAS-systemenes mulighet for annotering eller memoisering som kan bygges ut til modeller i et hierarkisk begrepssystem. Begrepssystemet (ontologien) jeg foreslår å bruke er som nevnt i avsnitt 4.2.3 CIDOK-CRM.

4.3.2 Ta vare på informasjon

Begrunnelsessystemet er et eksempel på at man tar vare på informasjon når man har den tilgjengelig. Det er viktig å være klar over at dette ikke bare er en praktisk måte å organisere arbeidet på, det er et spørsmål om man senere har informasjon eller ikke. Det er således ikke å sammenligne med et valg om man merker opp informasjonskategorier i en tekst eller ikke, for eksempel hvorvidt man skriver inn harde linjeskift fra originalen eller om man setter tagger rundt stedsnavn. Hvis man merker opp en tekst og unnlater å markere linjeskift eller personnavn, er det helt greit å gjøre den oppmerkingen senere så lenge man har tilgang til originalen. Men den typen tilleggsinformasjon man kan skrive inn i begrunnelsessystemet er borte senere, fordi det ikke er snakk om å uttrykke attributter i teksten, men å uttrykke tildelerens tanker i det et valg blir tatt.

For mange av de valgene som ble gjort i denne oppgaven, kan man ved å gjenta prosessen, det vil som oftest si at man leser teksten på nytt, lett kunne rekonstruere det som ble lagt inn som grunn, f.eks. at et avsnitt har flere talere. Men i andre tilfelle, dette gjelder for eksempel hvilken kategori en personbeskrivelse kobles til, ligger det gjerne vurderinger som ikke kan tenkes ut på nytt med samme resultat. Dersom slike grunner ikke tas vare på, vil de forsvinne, og kan ikke rekonstrueres.

Det kunne vært nyttig når man skal gjøre et valg å få se de begrunnelser som er lagt inn på tidligere valg av samme type. Det kunne gjøre det lettere å få til en enhetlig praksis. Det har jeg imidlertid ikke implementert i programsystemet i denne oppgaven.

Et problem er at man ikke vet hva man skal ta vare på. Det er ikke mulig å begrunne alt, vi kan ikke engang gi begrunnelser for de fleste valg i en normal arbeidssituasjon fordi det ville tatt for lang tid. Men problemet med grenser for begrunnelser er større, fordi det er aktuelt også på et annet nivå. I hvilken grad skal bakgrunnen for mekaniske valg knyttes til begrunnelsessystemet?

Jeg har f.eks. vurdert å legge inn en begrunnelse for hvert sidetall som legges inn på noder i det DOM-lignende treet, men reglene for innlegging er så enkle at dette er ikke gjort. Det ville representert et stort antall begrunnelser som har liten verdi, da man heller kan henvise brukeren til de generelle regelbeskrivelsene. Men hvordan skal en slik henvisning gjøres?

Begrunnelsessystemet må på en eller annen måte knyttes til ikke-begrunnede rutiner i systemet slik at de valgene som er begrunnet i begrunnelsessystemet sees i sammenheng med de ikke-begrunnede valgene. Det betyr at når systemet

trekker en slutning må reglene for denne slutningen, som er uttrykt i Lisp i programmet, kunne uttrykkes på en tydelig måte sammen med resultatet av slutningen.

4.3.3 Nye publiseringformer

Selv om en eksport til TEI og CIDOK-CRM kan legge grunnlag for en publisering, er det mange andre former for publisering man kan tenke seg. Denne oppgaven er en publisering av programsystemet og konklusjonene av analysene, sammen med en argumentasjon for konklusjonene basert på kildene. Formen på oppgaven er preget av hvordan man i universitetene har utviklet krav til og sendvane for publisering av hovedoppgaver. Dette innebærer at selv om jeg i oppgaven skriver om bedre referansesystemer enn de som brukes i trykte tekster, er oppgaven selv en tradisjonell trykt tekst. Å lagre begrunnelser og pekere løser ikke i seg selv det grunnleggende problemet: Hvordan unngå at leseren bare får tilgang til en tabell uten lenker til underliggende data? Hvorfor lage et begrunnelsessystem dersom det ikke brukes? Begrunnelsessystemet er riktignok brukt i arbeidet, og muligheten til bedre dokumentasjon av arbeidet mens man er i gang med det er viktig for forskeren i arbeid. Men jeg håper på noe mer.

Et annet problem er kanskje vel så stort: Gitt at et bedre etterprøvbart system lages, vil brukeren *bruke* tilgangen? Man ser jo hvordan fotnoter, som man gjerne mener skal være der for å brukes, ofte fungerer legitimerende, selv om de ikke blir brukt som referanser av leseren:

> Long lists of earlier books and articles and strings of coded references to unpublished documents supposedly prove the solidity of the author's research by rendering an account of the sources used. In fact, however, only the relatively few readers who have trawled their nets through the same waters can identify the catch in any given set of notes with ease and expertise. For most readers, footnotes play a different role. In a modern, impersonal society, in which individuals must rely for vital services on other whom they do not know, credentials perform what used to be the function of guild membership or personal recommendations: they give legitimacy. (Grafton 1997, s. 7)

Et av hovedproblemene er informasjonsmengden. En enkelt analyse kan basere seg på mange tusen ord, flere hunder avsnitt og flere titalls personer. Hvordan er det mulig å bruke dette for en leser, selv om det er tilgjengelig? Dersom pekere fra analysedata tilbake til kildetekst skal kunne publiseres på en meningsfull måte, må informasjonsmengden håndteres på en måte som ikke oversvømmer leseren, samtidig som den kildekritiske pekerrekka holdes intakt fra manuskript til kritisk analyse.

Et eksempel på hvordan dette kan gjøres er følgende: Alle avsnitt som ligger til grunn for en påstand om at bønder snakker slik-og-slik kan listes opp. Brukeren kan så for hvert avsnitt be om å få vist fram bakgrunnen for at dette

avsnittet er tilknyttet en bonde som taler, inkludert begrunnelser skrevet inn av brukeren. Er en slik metode god nok, eller må større deler av funksjonaliteten i programsystemet publiseres for at det skal bli brukbart for en leser? Dersom man gjør en slik publisering, betyr det at leseren blir en bruker som kan manøvrere i systemet og se hvordan konklusjonene henger sammen med, eller ikke henger godt nok sammen med, kildene? Dette bør i tilfelle kunne gjøres i form av en nettbasert hypertekstutgave.

Spørsmålet blir således: Vil også digitale utgaver kun inneholde konklusjonene, og slik bli utgaver som er nokså like denne oppgaveteksten, eller åpner det for en publisering som kan gi noe mer i kobling til kildene? I denne oppgaven er det vist hvordan viktig informasjon som er tilgjengelig i en arbeidsprosess kan lagres i maskinelt støttet arbeid med digitalt kildemateriale. Men å lage gode måter å publisere slik informasjon på må det arbeides videre med.

Bibliografi

[Bradley 1991] Bradley, John: "TACT Design." I: *CCH Working Papers* vol. 1 (1991). URL: http://www.chass.utoronto.ca/epc/chwp/bradley/ — lenke sjekket 2004-10-03.

[Bradley 2000] Bradley, John. "Tools to augment scholarly activity: an architecture to support text analysis." Informatica Umanistica: Filosofia e Risorse Digitali, Bologna, Italy, 2000. URL: http://pigeon.cch.kcl.ac.uk/docs/papers/bologna/ — lenke sjekket 2004-08-18.

[Buchanan 1989] Buchanan, Bruce G. og Reid G. Smith: "Fundamentals of Expert Systems." Ss. 149–192 i: *The Handbook of Artificial Intelligence. Volume IV* / Avron Barr, Paul R. Cohen og Edward A. Feigenbaum (red.). Reading, 1989.

[Burke 2000] Burke, Peter: *A social history of knowledge : from Gutenberg to Diderot.* Cambridge, 2000.

[Busa 1998] Busa, Robert SJ: "Concluding a life's safari from punched cards to World Wide Web." Ss. 3–11 i: *The Digital Demotic: A Selection of Papers from DRH97 (Digital Resources for the Humanities).* / London, 1998.

[CIDOK CRM 2004] *Definition of the CIDOC Conceptual Reference Model. Version 4.0.* / Nick Crofts, Martin Doerr, Tony Gill, Stephen Stead, Matthew Stiff (red.) Produced by the ICOM/CIDOC Documentation Standards Group, continued by the CIDOC CRM Special Interest Group. April 2004

[Czmiel 2004] Czmiel, Alexander: "XML for Overlapping Structures (XfOS) using a non XML Data Model." Ss. 48–49 i: *ALLC/ACH 2004. Computing and Multilingual, Multicultural Heritage. Conference abstracts.* Göteborg, 2004.

[DN-web] *Avansert søk i Diplomatarium Norvegicum.* URL: http://www.dokpro.uio.no/dipl_norv/diplom_felt.html — lenke sjekket 2004-04-05.

[Dokpro WWW] *Vevsidene til Dokumentasjonsprosjektet.* URL: http://www.dokpro.uio.no/ — lenke sjekket 2004-08-12.

[Dokpro 1998] *Dokumentasjonsprosjektet. Sluttrapport 1992–1997.* / Christian-Emil Ore og Nina Kristiansen (red.). Oslo, 1998.

[DOM 2004] *Document Object Model (DOM) Level 3 Core Specification.* Version 1.0. W3C Recommendation 07 April 2004. URL: http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/ — lenke sjekket 2004-08-19.

[Eide 1998 A] Eide, Øyvind og Tor Sveum: *Dokumentasjonsprosjektet ved Universitetsbiblioteket i Tromsø. Rapport.* Tromsø, 1998.

[Eide 1998 B] Eide, Øyvind: "Re-integrasjon av kildemateriale." Ss. 7–20 i: *Human IT* 1, 1998.

[Eide 2004] Eide, Øyvind, Jon Holmen og Anne Birgitte Høy-Petersen: "Between the Book and the Exhibition : Creating Archaeological Presentations Based on Database Information." *Proceedings of the CAA 2004 conference, Prato, Italy* (under utgivelse).

[Fielding 2002] Fielding, Nigel G.: "Automating the ineffable: Qualitative software and the meaning of qualitative research." Ss. 161–178 i: *Qualitative Research in Action* / red.: Tim May. London, 2002.

[Firth 1957] Firth, J.: "A Synopsis of Linguistic Theory 1930-1955" Ss. 1–32 i: *Studies in Linguistic Analysis.* Oxford, 1957.

[Gaski 1995] Gaski, Harald: "Samisk kultur i går — i dag — i morgen." I: *P2-akademiet. D.* Oslo, 1995.

[GNU CLISP 2001] *GNU CLISP 2.27* [dataprogram]. Operativsystem: Solaris. Datert 17. juli 2001.

[Grafton 1997] Grafton, Anthony: *The Footnote : A curious history.* London, 1997.

[Graham 1996] Graham, Paul: *ANSI Common Lisp.* Upper Saddle River, 1996

[Granaas 2002] Granås, Peggy Helen: *Statsmakt og misjon : organiseringen av finnemisjonen i Nord-Norge i første halvdel av 1700-tallet.* Hovedoppgave i historie - Norges teknisk-naturvitenskapelige universitet, 2002.

[Hansen 2004] Hansen, Lars Ivar og Bjørnar Olsen: *Samenes historie : fram til 1750.* Oslo, 2004

[Holmes 1991] Holmes, David I.: "A multivariate technique for author attribution and its application to the analysis of Mormon scripture and related texts." Ss. 12–22 i: *The Journal of History and Computing* / 3(1991), nr. 1.

[HTML 1999] *HyperText Markup Language (HTML) 4.01 Specification.* W3C Recommendation 24 December 1999. URL: http://www.w3.org/TR/html4/ — lenke sjekket 2004-09-28.

[ISO-8879] *ISO 8879:1986. Information processing — Text and office systems — Standard Generalized Markup Language (SGML).* 1986.

[Jordal 2004] Jordal, Ellen, Jon Holmen, Stein A. Olsen og Christian-Emil Ore: "From XML-tagged Acquisition Catalogues to an Event-based Relation Databases." *Proceedings of the CAA 2004 conference, Prato, Italy* (under utgivelse).

[Kjeldstadli 1999] Kjeldstadli, Knut: *Fortida er ikke hva den en gang var.* 2. utg. Oslo, 1999.

[Kleio WWW] *Kleio on the web.* URL: http://wwwuser.gwdg.de/˜mthalle2/ — lenke sjekket 2004-10-15.

[Lancashire 1986] Lancashire, Ian: "Concordance programs for Literary Analysis." Ss. 54–61 i: *SIGCUE Outlook.* 19 nr. 1/2, 1986.

[Luhn 1966] Luhn, H.P.: "Keyword-in-Context Index for Technical Literature (KWIC Index)." Ss. 159–167 i: *Readings in Automatic Language Processing* / red.: David G. Hays. New York, 1966.

[McCarty 1993] McCarty, Willard: "Handmade, Computer-Assisted, and Electronic Concordances of Chaucer". Ss. 49-65 i: *Computer-Assisted Chaucer Studies* / red.: Ian Lancashire. Toronto, 1993. URL: http://www.kcl.ac.uk/humanities/cch/wlm/essays/concordance/ — lenke sjekket 2004-08-24

[Myrhaug 1997] Myrhaug, May-Lisbeth: *I modergudinnens fotspor : samisk religion med vekt på kvinnelige kultutøvere og gudinnekult.* Oslo, 1997.

[NARA 2000] "The Soundex Indexing System." Basert på brosjyren "Using the Census Soundex," General Information Leaflet 55 (Washington, DC: National Archives and Records Administration, 1995). URL: http://www.archives.gov/research_room/genealogy/census/soundex.html — lenke sjekket 2004-10-05.

[Norvig 1992] Norvig, Peter: *Paradigms of Artificial Intelligence Programming : Case Studies in Common Lisp.* San Mateo, 1992.

[NOU 1985:14] *NOU 1985:14. Samisk kultur og utdanning.* Oslo, 1985

[Oakes 1998] Oakes, Michael P.: *Statistics for Corpus Linguists.* Edinburgh, 1998.

[Olsen 1988] Olsen, Mark og Louis-Georges Harvey: "Computers in Intellectual History: Lexical Statistics and the Analysis of Political Discourse." Ss. 449–464 i: Journal of Interdisciplinary History, Vol. 18, nr. 3, 1988.

[Oracle 2002] *Oracle9i XML API Reference - XDK and Oracle XML DB.* Release 2 (9.2). Part Number A96616-01. 2002.

[Ore 2002] Ore, Christian-Emil Smith. "Datateknologi og gamle breve, Diplomatarium Norvegicum i elektronisk form." Ss. 167–180 i: *Ny väg till medeltidsbreven: Riksarkivet i Sverige.* Stockholm, 2002

[Pfeifer 2000] Pfeifer, Rolf og Christian Scheier: *Understanding Intelligence.* Cambridge, 2000.

[Pollan 1993] Pollan, Brita: *Samiske sjamaner : religion og helbredelse.* Oslo, 1993.

[Renbeitekommissionen 1909 2] *Dokumenter angaaende flytlapperne m.m. : samlede efter renbeitekommissionens opdrag* / af J. Qvigstad og K.B. Wiklund. Bind 2. Kristiania, 1909.

[Rockwell 2003] Rockwell, Geoffrey: "What is Text Analysis, Really?" Ss. 209–219 i: *Literary and Lingustic Computing.* Vol. 18(2003), nr. 2.

[Rydving 1995] Rydving, Håkan: *The End of Drun-Time. Religious Change amoung the Lule Saami, 1670s–1740s.* 2. utg. Uppsala, 1995.

[Sandmo 1992] Sandmo, Erling: *Tingets tenkemåter. Kriminalitet og rettssaker i Rendalen, 1763–97.* Oslo, 1992.

[Schnitler 1962] Schnitler, Peter: *Major Peter Schnitlers grenseeksaminasjonsprotokoller 1742–1745. Bind 1* / ved Kristian Nissen og Ingolf Kvamen. Oslo, 1962.

[Schnitler 1929] Schnitler, Peter: *Major Peter Schnitlers grenseeksaminasjonsprotokoller 1742–1745. Bind 2* / ved J. Qvigstad og K. B. Wiklund. Oslo, 1929.

[Schnitler 1985] Schnitler, Peter: *Major Peter Schnitlers grenseeksaminasjonsprotokoller 1742–1745. Bind 3* / ved Lars Ivar Hansen og Tom Schmidt. Oslo, 1985.

[Sinclair 2003] Sinclair, Stéfan: "Computer-Assisted Readingt: Reconceiving Text Analysis." Ss. 175–183 i: *Literary and Linguistic Computing*, Vol. 18(2003), nr. 2.

[Sinclar 2004] Sinclair, Stéfan: "Text Analysis Markup Language (TAML)." S. 187 i: *ALLC/ACH 2004. Computing and Multilingual, Multicultural Heritage. Conference abstracts.* Göteborg, 2004.

[Smail 1999] Smail, Daniel Lord: *Imaginary cartographies : possession and identity in late medieval Marseille.* Ithaca, 1999.

[Sogner 1996] Sogner, Sølvi: *Aschehougs Norgeshistorie. Bind 6. Krig og fred : 1660-1780.* Oslo, 1996

[SPSS 2003] *SPSS 12.0.1* [dataprogram]. Operativsystem: Windows 2000. Datert 11. nov. 2003.

[Steen 1954] Steen, Adolf: *Samenes kristning og finnemisjonen til 1888.* (Avhandlinger utgitt av Egede-Instituttet ; 5) Oslo, 1954.

[Stewart 2004] Stewart, Larry: "Moll Flanders: Calculating Voice." Ss. 130–132 i: *ALLC/ACH 2004. Computing and Multilingual, Multicultural Heritage. Conference abstracts.* Göteborg, 2004.

[Stretton, 1997] Stretton, Tim: "Social historians and the records of litigation." Ss. 15–34 i: *Fact, fiction and forensic evidence. The potential of judicial sources for historical research in the early modern period.* Oslo, 1997.

[Strømstadtraktaten 1967] *Traktat om grensen mellom Norge og Sverige.* Undertegnet i Strømstad, 02-10-1751. Ss. 8–22 i: *Norges traktater. Bind 1, 1661–1944* / utarbeidet av det Kgl. norske utenriksdepartement. Oslo, 1967.

[Tanselle 1995] Tanselle, G. Thomas: "The Varieties of Scholarly Editing." Ss. 9–32 i: *Scholarly editing : a guide to research* / red: D.C. Greetham. New York, 1995.

[TEI 2002] *TEI P4: Guidelines for Electronic Text Encoding and Interchange. Text Encoding Initiative Consortium. XML Version* / C.M. Sperberg-McQueen og L. Burnard (red.) Oxford, 2002. URL: http://www.tei-c.org/P4X/ — lenke sjekket 2004-04-09.

[Thaller 1985] Thaller, Manfred: "Beyond Collecting. On the design and implementation of CLIO, a DBMS for the Historical Sciences." Ss. 328–334 i: *Data Bases in the Humanities and the Social Sciences 2.* Osprey, 1985.

[Thaller 1991] Thaller, Manfred: "The Historical Workstation Project." Ss. 149–162 i: *Computers and the Humanities* 25, nr. 1, 1991.

[Thorvaldsen 1996] Thorvaldsen, Gunnar: *Håndbok i registrering og bruk av historiske persondata.* Oslo, 1996.

[Thorvaldsen 1999] Thorvaldsen, Gunnar: *Databehandling for historikere.* Oslo, 1999.

[Tingbok WWW] Tingbokprosjektet ved Universitetet i Oslo, URL: http://www.hf.uio.no/hi/prosjekter/tingbok/ — lenke sjekket 2004-04-27.

[TUSTEP WWW] *Tübinger System von Textverarbeitungs-Programmen.* URL: http://www.uni-tuebingen.de/zdv/tustep/index.html — sjekket 2004-10-03.

[Vannebo 1984] Vannebo, Kjell Ivar: *En nasjon av skriveføre : om utviklinga fram mot allmenn skriveferdighet på 1800-tallet.* Oslo, 1984.

[XLISP-STAT 1999] *XLISP-STAT Release 3.52.20 (Beta)* [dataprogram]. Operativsystem: Solaris. Copyright (c) 1989-1999, by Luke Tierney.

[XML 2004] *Extensible Markup Language (XML) 1.0 (Third Edition)* / François Yergeau, Tim Bray, Jean Paoli, C. M. Sperberg-McQueen og Eve Maler (red.). W3C Recommendation 04 February 2004. URL: http://www.w3.org/TR/2004/REC-xml-20040204 — lenke sjekket 2004-04-09.

[XSLT 2003] *XSL Transformations (XSLT) Version 2.0.* W3C Working Draft 12 November 2003. URL: http://www.w3.org/TR/2003/WD-xslt20-20031112/ — lenke sjekket 2004-10-25

[Ystad 2000] Ystad, Vigdis: "Filologiens renessanse? Moderne edisjonsfilologi, hermeneutikk og tekstkritikk." Ss. 225–240 i: *Norsk litterær årbok.* Oslo, 2000.

Tillegg A

Kodelisting

A.1 sgml-tre

```
;;; Programmet baserer seg paa at SGML-dokumentet er velformet i XMLsk
;;; forstand.


;;;************************************************************
;;;
;;;     GENERELL SGML-DEL
;;;
;;;============================================================

;;============================================================
;; DEFINISJONER
;;------------------------------------------------------------

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Globale variable start                                      ;
;-------------------------------------------------------------;

(setq *denne* '()) ; Listevariabel som holder aktuell node i
                   ; legg-inn-ny-sgml og legg-inn-i-tre

(setq *buffer* "") ; Strengvariabel som holder det som til enhver tid
                   ; er i lesebufferet

(setq *innfil* '()) ; Holder filpekeren til inputfila

(setq *filslutt* nil) ; Boolsk verdi som angir om fila er lest til
                      ; slutten.

(setq *idnummer* 0) ; Numerisk variabel som teller opp hver nodes
                    ; unike id, brukes i legg-til-neste-barn og
                    ; legg-inn-resten-sgml

(setq *pnavnreg-idstart* 0) ; Numerisk variabel som passer på hvor
                            ; pnavn-registeret starter.
```

```
(setq *pnavnreg-idslutt* 0) ; Numerisk variabel som passer på hvor
                            ; pnavn-registeret slutter.

(setq *nodene* (make-hash-table)) ; Hashtabell som holder pekere til
                                  ; alle nodeobjektene indeksert på
                                  ; den unike id'en, brukes i
                                  ; lag-element, lag-pcdata og
                                  ; outputprosedyrer

(setq *tomme-elementer* '(SIDE SNR SPALTER SPALTE VSKILLE))

(setq *bokstaver* '"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
                   ÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÑÒÓÔÕÖØÙÚÛÜÝßàáâãäåæçèéêëìíîïñòóôõöøùúûüýÿ")

(setq *brukerdialog* '()) ; Settes til true for å få spørmål når
                          ; programmet trenger hjelp. Hvis ikke, gir
                          ; det opp.


;;; Fixer en uønsket default i CLISP, jfr. "Implementation Notes for
;;; GNU CLISP 2.31", kap. 22.1: Multiple Possible Textual
;;; Representations (URL: http://clisp.sourceforge.net/impnotes.html
;;; pr. 2003-09-30)
(setf *PPRINT-FIRST-NEWLINE* '())

;(setq *filtre* '"/home/oeide/hedvig/hf/hedvig/muspro-u1/oeide/utv/lisp/hoppg/")
(setq *filtre* '"~/utv/lisp/hoppg/")
                                  ; Rota i filtreet der programmer og filer ligger.

(setq *innfilnavn-tekst* (concatenate 'string
                                      *filtre*
                                      "testdata/schnitler-trimmet.sgml"))

(setq *innfilnavn-pnavn-reg* (concatenate 'string
                                          *filtre*
                                          "testdata/schnitl-474ff.sgml"))

(setq *utfilnavn* (concatenate 'string
                               *filtre*
                               "testdata/schnitler-ut.sgml"))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Nodeklassen                                                          ;
;----------------------------------------------------------------------;

(defclass node ()
  ((id :initarg :id :reader nodeid)
   (mamma :initarg :mamma :accessor nodens-mamma)
   (barneliste :initform '() :accessor nodebarn)
   (sidetall :initform 0 :accessor sidetall) ; Ikke helt generell
   (sidetalltype :initform '"" :accessor sidetalltype) ; Ikke helt generell
   (taler :initform '() :accessor nodens-taler))
  (:documentation "Hovedklasse for noder. Kun subklasser av bestemte
  typer brukes. id er en unik identifikator. mamma peker oppover i
  treet. barnliste er pekere til barna sortert i riktig
```

```
  rekkefølge. sidetall er hvilket sidetall noden hører til (en form
  for caching, da det kunne vært en funksjon). sidetalltype angir
  hvilken serie sidetall. Taler angir hvem som har sagt/skrevet avsnittet."))

(defclass element (node)
  ((navn :initarg :navn :reader elementnavn)
   (attributtliste :initarg :attributtliste :accessor elementattributtliste)
   (registerobjekt :initform '() :accessor registeroobjekt))
  (:documentation "Klasse for elementer. navn er elementnavnet som
    streng. attributtliste er alle attributtene i ei liste av
    navn-verdi-par, der navn og verdi er strenger. registerobjekt er
    objektet som representerer elementet i ett register."))

(defclass pcdata (node)
  ((innhold :initarg :innhold :reader pcdatainnhold))
  (:documentation "Klasse for PCDATA. innhold er selve strengen som tekst."))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Les fra fil                                                          ;
;----------------------------------------------------------------------;

(defun neste-tagg ()
  "Leverer neste tagg fra *buffer*, conset med evt. PCDATA før
  taggen. Fyller opp *buffer* fra input-strøm ved behov. Utdata:
  streng streng:         Pcdata før neste tagg og neste tagg
  tom-streng streng:     Neste tagg kommer rett etter forrige tagg
  streng tom-streng:     Ingen flere tagger, resten av fila som pcdata
  tom-streng tom-streng: Fil slutt"
  (let ((tagg-start (finn-tagg-start)))
    (if tagg-start ; Denne sjekken i finn-tagg-start i stedet?
        (let* ((pcdata-del (subseq *buffer* 0 tagg-start))
               (tagg-slutt (finn-tagg-slutt))
               (tagg-streng (subseq *buffer*
                                    (incf tagg-start)
                                    tagg-slutt)))
          (setq *buffer* (subseq *buffer* (incf tagg-slutt)))
          (cons pcdata-del tagg-streng))
        (if (fyll-buffer)
            (neste-tagg)
            (cons *buffer* '("")))))

(defun tagg-del (input)
  "Gir den delen av et par som er definert som taggnavn med attributter"
  (declare (cons input))
  (cdr input))

(defun pcdata-del (input)
  "Gir den delen av et par som er definert som pcdata"
  (declare (cons input))
  (car input))

(defun fyll-buffer ()
  "Henter mere data fra inputstrømmen *innfil* og legger i
   *buffer*. Returnerer #f når inputfila er tom."
  (let ((neste-linje (read-line *innfil* nil nil)))
    (if neste-linje
```

```
        (setq *buffer* (concatenate 'string *buffer*
                                    (string (code-char 10)) neste-linje))
        nil)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Behandling av SGML-tekst                                              ;
;-----------------------------------------------------------------------;

(defun finn-tagg-start ()
  "Finner posisjonen til neste < i *buffer*. Sjekker ikke om bufferet
   er tomt her, men i neste-tagg. Litt inkonsekvent i forhold til
   finn-tagg-slutt."
  (position #\< *buffer*))

(defun finn-tagg-slutt ()
  "Finner posisjonen til neste > i buffer. Fyller bufferet ved
   behov. Feil dersom det ikke finnes noen >."
  (let ((pos (position #\> *buffer*)))
    (if pos
        pos
        (if (fyll-buffer)
            (finn-tagg-slutt)
            (error (concatenate 'string
                                "Fantes ingen > i fila som passer med siste <: "
                                *buffer*))))))

(defun sluttaggp (streng)
  "Sjekker om parameteret har form av en sluttagg"
  (declare (string streng))
  (string= (subseq streng 0 1) "/"))

(defun finn-tagg-navn (streng)
  "Finner den delen av parametert som er taggnavnet"
  (declare (string streng))
  (let ((blank (position (code-char 32) streng)))
    (if blank
        (subseq streng 0 blank)
        streng)))

(defun plukk-attributt (streng)
  "Returnerer ei liste bestående av par med hvert attributtnavn og
   hver attributtverdi. Takler nå også manglende innkapsling, men ikke
   blanding av innkapslet og ikke."
  (declare (string streng))
  (if (tom-strengp streng)
      '()
      (let* ((streng (string-trim " " streng))
             (lik (position #\= streng))
             (anf1 (position #\" streng)))
        (if anf1 ; Går ut fra at det er innkapsling
            (let ((anf2 (position #\" streng :start (+ anf1 1))))
              (unless (= (- anf1 lik) 1) ; Slår til hvis man bruker '
                                         ; i stedet for " for å skille
                                         ; ut attributtverdien.
                (setq anf1 (position #\' streng))
                (setq anf2 (position #\' streng :start (+ anf1 1))))
              (let ((navn (subseq streng 0 lik))
```

```
                    (verdi (subseq streng (+ anf1 1) anf2)))
                (cons (cons navn verdi)
                      (plukk-attributt (subseq streng (incf anf2))))))
          ;;; Mangler innkapsling
          (let* ((start (+ lik 1))
                 (blank (position (code-char 32) streng :start start))
                 (slutt (if (and blank (< blank taggslutt))
                            (- blank 1)
                          (length streng))))
            (let ((navn (subseq streng 0 lik))
                  (verdi (subseq streng start slutt)))
              (cons (cons navn verdi)
                    (plukk-attributt (subseq streng slutt)))))))))

(defun finn-attributtliste (streng)
  "Tar inn en full starttag og returnerer ei liste av attributter med verdier."
  (declare (string streng))
  (let ((blank (position (code-char 32) streng)))
    (if blank
        (plukk-attributt (subseq streng (incf blank)))
        '())))

(defun spes-taggp (tagg)
  "Sjekker om taggen er en spesialform (doctype, dec, kommentar, ...)
   dvs. om den starter med <!"
  (string= (subseq tagg 0 1) "!"))

(defun empty-elementp (tagg)
  "Sjekker om tagen representerer et tomt element. Bør evt. erstattes
   av en DTD-parsing, slik den står er den hardkodet DTD-avhengig."
  (find-if #'(lambda (x) (string-equal (string-upcase tagg) x)) *tomme-elementer*))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Behandling av SGML-noder                                                  ;
;---------------------------------------------------------------------------;

(defun finnes-element-i-subtre (treliste elementnavn)
  "Sjekker om elementnavn finnes som node i subtrærne i liste. Hvis ja
   returneres ei liste med id'en til første node som finnes med navnet."
  (declare (cons treliste) (string elementnavn))
  (unless (null treliste)
    (if (and (typep (car treliste) 'element)
             (equal (elementnavn (car treliste)) elementnavn))
        (car treliste)
      (finnes-element-i-subtre (concatenate 'cons (nodebarn (car treliste))
                                            (cdr treliste)) elementnavn))))

(defun finnes-tekst-i-subtre (treliste tekst)
  "Sjekker om tekst finnes som tekst i en pcdata-node i subtrærne i
   liste. Hvis ja returneres ei liste med id'en til første node som
   finnes med navnet."
  (declare (cons treliste) (string tekst))
  (unless (null treliste)
    (if (and (typep (car treliste) 'pcdata)
             (search tekst (pcdatainnhold (car treliste))))
        (car treliste)
```

```
      (finnes-element-i-subtre (concatenate 'cons (nodebarn (car treliste))
                                            (cdr treliste)) tekst))))

(defun finn-attributtpar (attributtliste navn)
  "Finner attributtet med gitt navn i attributtlista og returnerer
   paret. Hvis den ikke finnes, returneres false."
  (declare (cons attributtliste) (string navn))
  (unless (null attributtliste)
    (find-if #'(lambda (par) (string-equal navn (car par))) attributtliste)))

(defun legg-til-attributt (node attributtpar)
  "Legger til et attributtpar (navn og verdi) på gitt
   elementnode. Dersom attributtet finnes fra før, oppdateres verdien."
  (if (typep node 'element)
      (let ((attributtpar-gammelt (finn-attributtpar (elementattributtliste node)
                                                     (car attributtpar))))
        (if attributtpar-gammelt
            (setf (elementattributtliste node)
                  (substitute-if attributtpar
                                 #'(lambda (par) (string-equal (car attributtpar)
                                                               (car par)))
                                 (elementattributtliste node)))
          (setf (elementattributtliste node) (cons attributtpar
                                                   (elementattributtliste node)))))
    (error "legg-til-attributt: Ikke elementnode: " node)))

(defun lag-attributtverdier-av-nodeid ()
  "Legger inn nodeide-verdien som attributt på alle elementer."
  (maphash (lambda (id node)
             (if (typep node 'element)
                 (legg-til-attributt node (cons '"ID" (format nil "~D" (nodeid node))))))
           *nodene*)
  (format t "~%Lagt inn id som attributter..."))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Tre-arbeid                                                          ;
;---------------------------------------------------------------------;

(defun lag-element (elementnavn attrib-liste mamma id)
  (declare (string elementnavn) (fixnum id))
  (setf (gethash id *nodene*) (make-instance 'element :navn elementnavn
                                                      :attributtliste attrib-liste
                                                      :mamma mamma :id id)))

(defun lag-pcdata (streng mamma id)
  (declare (string streng) (fixnum id))
  (setf (gethash id *nodene*) (make-instance 'pcdata :innhold streng
                                                     :mamma mamma :id id)))

(defun enslig-trep (tre)
  (null (nodebarn tre)))

(defun legg-til-neste-barn (tre elementnavn attrib-liste type)
  "Legger en node til i subtreet under tre og returnerer den nye noden"
  (if (null tre)
      (lag-element elementnavn attrib-liste '() (incf *idnummer*))
```

```
      (let ((ny-node
             (case type ('pcdata (lag-pcdata elementnavn tre
                                             (incf *idnummer*)))
                   ('element (lag-element elementnavn attrib-liste tre
                                          (incf *idnummer*)))
                   (otherwise
                    (error
                     (concatenate
                      'string
                      "legg-til-neste-barn: argumentet type maa vaere pcdata eller element: "
                      type))))))
        (when (= (mod *idnummer* 10000) 0)
          (format t "~A~D~%" "Antall noder: " *idnummer*))
        (setf (nodebarn tre) (append (nodebarn tre) (cons ny-node '())))
        ny-node)))


(defun sett-inn-node (ny-node ny-mamma forste-barn siste-barn)
  "Setter inn en ny node under noden ny-mamma. Alle barna til ny-mamma
   fra forste-barn - fra første barn om denne er null - til siste-barn
   - siste barn dersom denne er null - legges under den nye
   noden. Barnelista til ny-mamma skal etter operasjonen således være
   <barna fram til forste-barn> ny-node <barna etter siste-barn>. De øvrige
   nodene i barnelista legges som barneliste til ny-node."
  (let ((barneliste-mamma (nodebarn ny-mamma)))
    (if (and (or (not forste-barn) (find forste-barn barneliste-mamma))
             (or (not siste-barn) (find siste-barn barneliste-mamma)))
        (let* ((barneliste-mamma-start (if forste-barn
                                           (sosken-plass forste-barn)
                                         0))
               (barneliste-mamma-slutt (if siste-barn
                                           (sosken-plass siste-barn)
                                         (length barneliste-mamma)))
               (barneliste-ny-node (subseq barneliste-mamma
                                           barneliste-mamma-start
                                           barneliste-mamma-slutt))
               (barneliste-mamma-ny (concatenate 'cons
                                                 (subseq barneliste-mamma
                                                         0
                                                         barneliste-mamma-start)
                                                 (list ny-node)
                                                 (subseq barneliste-mamma
                                                         barneliste-mamma-slutt
                                                         (length barneliste-mamma)))))
          (setf (nodebarn ny-mamma) barneliste-mamma-ny)
          (setf (nodebarn ny-node) barneliste-ny-node)
          (mapc (lambda (bnode)
                  (setf (nodens-mamma bnode) ny-mamma))
                barneliste-mamma-ny)
          (mapc (lambda (bnode)
                  (setf (nodens-mamma bnode) ny-node))
                barneliste-ny-node))
      (error "sett-inn-node: Her er ikke forste-barn og siste-barn barn av ny-mamma!
Dette blir grådig feil."))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Les fra SGML-fil                                                    ;
```

```
;-------------------------------------------------------;


(defun legg-inn-ny-sgml ()
  "Legger inn et fullt SGML-tre basert på lesemekanikken. Sjekker
   først at innfila starter velformet, går så i gang med å lese resten av
   fila."
  (let* ((fil-par (neste-tagg))
         (fil-pcdata (pcdata-del fil-par))
         (fil-tagg (tagg-del fil-par)))
    (if (tom-strengp fil-tagg)
        (error (concatenate
                ’string
                "legg-inn-ny-sgml: Maa ha en starttagg i innfila: "
                fil-pcdata))
      (if (spes-taggp fil-tagg)
          (legg-inn-ny-sgml) ; Kan lage leamikk for å behandle spesialtagger også
        (let ((tagg (finn-tagg-navn fil-tagg))
              (attributtliste (finn-attributtliste fil-tagg)))
          (if (or (sluttaggp tagg) (empty-elementp tagg))
              (error
               (concatenate
                ’string
                "legg-inn-ny-sgml: Maa starte med starttagg i et element med innhold: "
                fil-tagg))
            (progn
              (setq *denne* (legg-til-neste-barn *rot* tagg attributtliste ’element))
              (legg-inn-resten-sgml)))))))))

(defun legg-inn-resten-sgml ()
  "Kjører innlegging av en og en tagg til fila er tom. Bruker do for å
   unngå å fylle stakken med rekursive kall. legg-inn-i-tre setter
   *filslutt* når fila er tom."
  (do ()
      (*filslutt* (format t "~A~D~%" "Sum antall noder: " *idnummer*))
    (legg-inn-i-tre)))

(defun legg-inn-i-tre ()
  "Holder oversikt over posisjon i treet med den globale variabelen
   *denne*. Gaar ut fra at alle sluttagger kommer paa riktig sted,
   dvs. velformet i XMLsk forstand."
  (let* ((fil-par (neste-tagg))
         (fil-pcdata (pcdata-del fil-par))
         (fil-tagg (tagg-del fil-par)))
    (if (tom-strengp fil-tagg) ; Slutt på fila
        (progn
          (unless (tom-strengp fil-pcdata)
            (legg-til-neste-barn *denne* fil-pcdata ’() ’pcdata))
          (setq *filslutt* 1)) ; Litt hakkete, kanskje..?
        (progn
          (unless (tom-strengp fil-pcdata)
            (legg-til-neste-barn *denne* fil-pcdata ’() ’pcdata))
          (unless (spes-taggp fil-tagg)
            (let ((tagg (finn-tagg-navn fil-tagg))
                  (attributtliste (finn-attributtliste fil-tagg)))
;;; Normaliseringen garanterer at kun tomme elementer kan vaere uten sluttagg.
                                        ; if starttag: gaa innover
                                        ; if sluttagg: fall ut
```

```
             (if (sluttaggp tagg)
                                  ; Her gaar vi ut fra at det er
                                  ; siste aapnede element som
                                  ; avsluttes.
                 (setq *denne* (nodens-mamma *denne*))
;;; Ikke sluttagg, så vi må sjekke om elementet er tomt
                 (if (empty-elementp tagg)
                     (setq *denne*
                           (nodens-mamma (legg-til-neste-barn
                                          *denne*
                                          tagg
                                          attributtliste
                                          'element)))
                     (setq *denne* (legg-til-neste-barn *denne*
                                                        tagg
                                                        attributtliste
                                                        'element)))))))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Output                                                              ;
;---------------------------------------------------------------------;

(defun skriv-attributter (attrliste)
  "Skriver ut attributtliste i SGML-format."
  (if (or (not attrliste) (null attrliste))
      ''""
    (if (position #\" (cdar attrliste))
        (concatenate 'string  ; Inneholder " i attributtverdi, bruker
                              ; ' for å skille den ut.
                     '" "
                     (caar attrliste)
                     '"=\'"
                     (cdar attrliste)
                     "\'"
                     (skriv-attributter (cdr attrliste)))
      (concatenate 'string
                   '" "
                   (caar attrliste)
                   '"=\""
                   (cdar attrliste)
                   "\""
                   (skriv-attributter (cdr attrliste))))))

(defun skriv-subtre-som-pcdata (rotnode &optional (ekskluderes '()))
  "Skriver ut alle pcdata i subtreet som starter med rotid-noden. Bør
   samle opp alle strengene og slå dem sammen før utskrift. Tar ikke
   hensyn til ekskluderes ennå."
  (defun skriv-subtre-som-pcdata-rekursiv (node ekskluderes)
    (when node
      (when (typep node 'pcdata)
        (princ (pcdatainnhold node)))
      (mapc #'skriv-subtre-som-pcdata-rekursiv
            (nodebarn node)
            (make-list (list-length (nodebarn node))
                       :initial-element ekskluderes))))
  (skriv-subtre-som-pcdata-rekursiv rotnode ekskluderes))
```

```
(defun finn-pcdata-fra-tre (nodeliste &key (ikke-noder '()))
  "Returnerer en tekst som er konkatingeringen av alle pcdata i
   subtreet. Innparameter-noden må pakkes i ei liste. Hvis ikke-noder er
   ikke-tom, returneres elementer av typer oppgitt som starttagger."
  (cond ((null nodeliste)
         '"")
        ((typep (car nodeliste) 'element)
         (let ((elementnavn (elementnavn (car nodeliste))))
           (if (find elementnavn ikke-noder :test #'equal-case)
               (concatenate 'string
                            (string-upcase elementnavn)
                            (finn-pcdata-fra-tre (cdr nodeliste)
                                                 :ikke-noder ikke-noder))
               (concatenate 'string
                            (finn-pcdata-fra-tre
                             (append (nodebarn (car nodeliste)) (cdr nodeliste))
                             :ikke-noder ikke-noder)))))
        ((typep (car nodeliste) 'pcdata)
         (concatenate 'string
                      (pcdatainnhold (car nodeliste))
                      (finn-pcdata-fra-tre
                       (append (nodebarn (car nodeliste)) (cdr nodeliste))
                       :ikke-noder ikke-noder)))
        (t
         (error "finn-pcdata-fra-tre: feil nodetype."))))

(defun finn-pcdata-fra-til (fra til &key ikke-noder teller)
  "Returnerer en tekst som er konkatingeringen av alle pcdata i nodene
   fra fra til til.Hvis ikke-noder er ikke-tom, returneres elementer av
   typer oppgitt som starttagger."
  (unless teller
    (setq teller 0))
  (let* ((nodenr (+ fra teller))
         (node (finn-node nodenr)))
    (cond ((> nodenr til)
           '"")
          ((typep node 'pcdata)
           (concatenate 'string
                        (pcdatainnhold node)
                        (finn-pcdata-fra-til fra
                                             til
                                             :teller (+ teller 1)
                                             :ikke-noder ikke-noder)))
          (t
           (if (null ikke-noder)
               (finn-pcdata-fra-til fra til :teller (+ teller 1)
                                    :ikke-noder ikke-noder)
               (let ((elementnavn (elementnavn node)))
                 (if (find elementnavn ikke-noder :test #'equal-case)
                     (concatenate 'string
                                  (string-upcase elementnavn)
                                  (finn-pcdata-fra-til
                                   fra til
                                   :teller (- (finn-node-id-etter-denne node) fra)
                                   :ikke-noder ikke-noder))
                   (finn-pcdata-fra-til fra til
```

```
                                        :teller (+ teller 1)
                                        :ikke-noder ikke-noder)))))))

(defun finn-pcdata-ord-fra-noder (fra til)
  "Skriver en liste av alle ordene i pcdata i avsn i nodene med id fra
   til. Skalerer."
  (dotimes (teller (- til fra))
    (let ((node (finn-node (+ teller fra))))
      (when (and (typep node 'element)
                 (equal (string-downcase (elementnavn node)) '"avsn"))
        (format t "~{~%~A~}" (hent-ord (finn-pcdata-fra-tre (list node)))))))
  (format t "~%~%"))

(defun finn-pcdatanode-med-txt (nodeliste tekst)
  "Returnerer første node i subtrærne i nodeliste som inneholder
   tekst. Normaliserer teksten i pcdata-nodene ved å skifte ut linjeskift
   med blank."
  (cond ((null nodeliste)
         '())
        ((typep (car nodeliste) 'element)
         (finn-pcdatanode-med-txt (append (nodebarn (car nodeliste))
                                          (cdr nodeliste)) tekst))
        ((typep (car nodeliste) 'pcdata)
         (if (search tekst (substitute (code-char 32) (code-char 10)
                                       (pcdatainnhold (car nodeliste))))
             (car nodeliste)
           (finn-pcdatanode-med-txt (append (nodebarn (car nodeliste))
                                            (cdr nodeliste)) tekst)))
        (t
         (error "finn-pcdatanode-med-txt: feil nodetype."))))

(defun finn-noder-i-subtre (nodeliste navn)
  "Returnerer ei liste med alle elementer av type navn i
   subtreet. Innparameter-noden må pakkes i ei liste."
  (cond ((null nodeliste)
         '())
        ((typep (car nodeliste) 'element)
         (append (if (equal (string-downcase (elementnavn (car nodeliste)))
                            (string-downcase navn))
                     (list (car nodeliste))
                   '())
                 (finn-noder-i-subtre (append (nodebarn (car nodeliste))
                                              (cdr nodeliste)) navn)))
        ((typep (car nodeliste) 'pcdata)
         (append '()) (finn-noder-i-subtre (append (nodebarn (car nodeliste))
                                                   (cdr nodeliste)) navn))
        (t
         (error "finn-noder-i-subtre: feil nodetype."))))

(defun finn-node-ider (navn &optional (fra 1) (til *idnummer*))
  "Returnerer ei liste med idnumre paa alle noder som representerer
   elementer av typen nodenavn. Ikke case-sensitiv. Mer effektivt enn å
   kjøre finn-noder-i-subtre på hele treet."
  (let ((elementid '()))
    (do* ((id fra (+ id 1))
          (node (gethash id *nodene*) (gethash id *nodene*)))
         ((or (not node)
```

```
            (and til (< til id)))
          (reverse elementid))
         (when (and (typep node 'element)
                    (equal (string-downcase (elementnavn node))
                           (string-downcase navn)))
           (setq elementid (cons id elementid))))))

(defun finn-node-id-etter-denne (node)
  "Finner ide'en til noden rett etter denne."
  (let ((sos (sosken-rett-etter node))
        (mamma (nodens-mamma node)))
    (cond (sos
           (nodeid sos))
          (mamma
           (finn-node-id-etter-denne mamma))
          (t
           (error "Ingen node etter...?")))))

(defun finn-node (id)
  "Returnerer peker til objektet med gitt id"
  (gethash id *nodene*))

(defun finn-node-over (node navn)
  "Finner første noden over node i treet som representerer et element
   av typen navn."
  (cond ((not node)
         (error "finn-node-over: Ut av treet!"))
        ((and (typep node 'element)
              (equal-case (elementnavn node) navn))
         node)
        (t
         (finn-node-over (nodens-mamma node) navn))))

(defun soskenflokk (node)
  "Returnerer ei liste med noder som er søskenflokken node inngår i."
  (let ((mor (nodens-mamma node)))
    (if mor
        (nodebarn mor)
      (error "Soskenflokk: Fant ikke mor til noden."))))

(defun sosken-plass (node)
  "Returnerer nodes plass i søskenflokken. 0-indexert."
  (let ((pos (position node (soskenflokk node))))
    (if pos
        pos
      (error (concatenate 'string "Sosken-plass: intern feil: "
                          (string (nodeid node)) node)))))

(defun sosken-foer (node)
  "Returnerer alle søsken som kommer før node."
  (let ((sosken (soskenflokk node))
        (nodepos (sosken-plass node)))
    (subseq sosken 0 nodepos)))

(defun sosken-rett-foer (node)
  "Returnerer siste søsken rett før node."
  (let ((sosken-foer (sosken-foer node)))
```

```
    (if (null sosken-foer)
        '()
      (car (last sosken-foer)))))

(defun sosken-etter (node)
  "Returnerer alle søsken som kommer etter node."
  (let ((sosken (soskenflokk node))
        (nodepos (sosken-plass node)))
    (subseq sosken (+ nodepos 1))))

(defun sosken-rett-etter (node)
  "Returnerer første søsken rett etter node."
  (let ((sosken-etter (sosken-etter node)))
    (if (null sosken-etter)
        '()
      (car sosken-etter))))

(defun sosken-rett-etter-eller-over (node stoppnavn)
  "Returnerer første søsken rett etter node eller, hvis ingen søsken
etter, søsken etter forfedre. Stopper ved som representerer elementer
av typen stoppnavn."
  (let ((sosken-etter (sosken-etter node)))
    (if (null sosken-etter)
        (let ((mamma (nodens-mamma node)))
          (if (and (typep mamma 'element)
                   (equal-case (elementnavn mamma) stoppnavn))
              '()
            (sosken-rett-etter-eller-over mamma stoppnavn)))
      (car sosken-etter))))

(defun finn-siste-nodeid (node)
  "Finner id'en til siste node i subtreet under node."
  (let ((sos (sosken-rett-etter node))
        (mamma (nodens-mamma node)))
    (cond (sos
           (- (nodeid sos) 1))
          (mamma
           (finn-siste-nodeid mamma))
          (t
           (idnummer)))))

(defun skriv-subtre-som-sgml (node &optional (strom *standard-input*))
  "Skriver ut subtreet med rot nodenummer"
  (when node
    (skriv-tre-som-sgml node strom)))

(defun skriv-subtre-elementene (nodenummer)
  "Skriver ut subtreet med rot nodenummer"
  (let ((subtre (gethash nodenummer *nodene*)))
    (when subtre
      (skriv-elementene-med-innrykk subtre))))

(defun skriv-subtre-elementene-pcdata (nodenummer)
  "Skriver ut subtreet med rot nodenummer"
  (let ((subtre (gethash nodenummer *nodene*)))
    (when subtre
      (skriv-elementene-og-pcdata-med-innrykk subtre))))
```

```
(defgeneric skriv-tre-som-sgml (tre &optional strom elementer)
  (:documentation "Skriver ut rotnoden basert paa om det er element
    eller pcdata, deretter skriver den ut hvert av barna. Siden rotnoden
    er dummy, bør den fjernes fra utskrift."))

(defmethod skriv-tre-som-sgml ((tre element) &optional (strom *standard-input*)
                               (elementer '()))
  "Skriver ut ett element. Burde vel sjekke på tomt element for i
   tilfelle å fjerne sluttaggen"
  (when (or (not elementer)
            (find (string-downcase (elementnavn tre)) elementer :test #'equal))
    (princ (concatenate 'string "<" (string (elementnavn tre))
                        (skriv-attributter (elementattributtliste tre)) ">") strom))
  (unless (enslig-trep tre)
    (mapc #'skriv-tre-som-sgml (nodebarn tre)
          (make-list (list-length (nodebarn tre)) :initial-element strom)
          (make-list (list-length (nodebarn tre)) :initial-element elementer)))
  (unless (empty-elementp (elementnavn tre))
      (when (or (not elementer)
            (find (string-downcase (elementnavn tre)) elementer :test #'equal))
        (princ (concatenate 'string "</" (string (elementnavn tre)) ">") strom)))
  nil)

(defmethod skriv-tre-som-sgml ((tre pcdata)
                               &optional (strom *standard-input*) (elementer '()))
  "Skriver ut pcdata."
  (princ (string (pcdatainnhold tre)) strom)
  (unless (enslig-trep tre)
    (mapc #'skriv-tre-som-sgml (nodebarn tre)
          (make-list (list-length (nodebarn tre)) :initial-element strom)
          (make-list (list-length (nodebarn tre)) :initial-element elementer)))
  nil)

(defun skriv-elementene-med-innrykk (tre &optional (innrykk 0))
  "Skriver ut alle elementene med ett innrykk for hvert nivaa."
  (when tre
    (progn
      (when (typep tre 'element)
        (format t "~A~D~A~%"
                (concatenate 'string
                             (make-string innrykk :initial-element (code-char 32))
                             (elementnavn tre)
                             " [")
                (nodeid tre)
                "]"))
      (mapc #'skriv-elementene-med-innrykk (nodebarn tre)
            (make-list (list-length (nodebarn tre))
                       :initial-element (incf innrykk))))))

(defun skriv-elementene-og-pcdata-med-innrykk (tre &optional (innrykk 0))
  "Skriver ut alle elementene med ett innrykk for hvert nivaa. Pcdata
   tas med med alle linjeskift erstattet av blanke."
  (when tre
    (progn
      (if (typep tre 'element)
          (format t "~A~D~A~%"
```

```
                (concatenate 'string
                             (make-string innrykk :initial-element (code-char 32))
                             (elementnavn tre)
                             " [")
                (nodeid tre)
                "]")
          (format t "~A~D~A~%"
                (concatenate 'string
                             (make-string innrykk :initial-element (code-char 32))
                             "("
                             (substitute (code-char 32) (code-char 10)
                                         (pcdatainnhold tre))
                             ") [")
                (nodeid tre)
                "]"))
      (mapc #'skriv-elementene-og-pcdata-med-innrykk (nodebarn tre)
            (make-list (list-length (nodebarn tre))
                       :initial-element (incf innrykk))))))

(defun finn-attributtverdi (attributtliste attributtnavn)
  "Finner attributtverdien til gitt attributtnavn, eller nil."
  (let ((attributt (assoc attributtnavn attributtliste :test #'string-equal)))
    (when attributt
      (cdr attributt))))

(defun skriv-sidetall ()
  "Skriver ut alle noder med sidetall. Ikke helt generell."
  (maphash (lambda (x y) (format t
                                 "~D~A~D~A~D~A~%"
                                 x
                                 ": "
                                 (sidetall y)
                                 " ("
                                 (sidetalltype y)
                                 ")"))
           *nodene*))


(defun skriv-tre-til-fil (utfilnavn)
  "Skriver hele treet til definert SGML-fil. Skriver ikke ut dec og
   <!... (dette legges jo ikke inn)"
  (setq *utfil* (open utfilnavn :direction :output))
  (mapc #'skriv-subtre-som-sgml
        (nodebarn *rot*)
        (make-list (list-length (nodebarn *rot*))
                   :initial-element *utfil*))
  (format *utfil* "~%")
  (clear-output *utfil*)
  (close *utfil*))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Globale variable slutt                                                    ;
;---------------------------------------------------------------------------;

(setq *rot* (legg-til-neste-barn '() '"sgml" '() 'element))
                                        ; Lager et dummy rotelement
                                        ; ytterst
```

A.2 parse-dokub

```
(load "sgml-tre-1.3.0.lsp")

;;;*************************************************************
;;;
;;;     SPESIELL DOKUB.DTD-DEL
;;;
;;;=============================================================

;;; Disse prosedyrene er hardkodet for den DTD'en vi bruker. Kan
;;; evt. parametrisere SIDE SNR og NR for å gjøre den mer generell,
;;; men likevel vil den være basert på at dokumentet har sider, at
;;; sideskille merkes (sider merkes mao. med milestones) og sidenumre
;;; finnes i et eget element et eller annet sted på sida (typisk
;;; øverst eller nederst, men det er ikke nødvendig).

;;=========================================================
;; DEFINISJONER
;;---------------------------------------------------------

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Globale variable DTD-lokale                                    ;
;----------------------------------------------------------------;

(setq *navnnummer* 0) ; Numerisk variabel som teller opp hver
                      ; navnenodes unike id

(setq *navnene* (make-hash-table)) ; Hashtabell som holder pekere til
                                   ; alle navnobjektene indeksert på
                                   ; den unike id'en.

(setq *pnavnoppsl* (make-hash-table :test #'equal)) ; Hashtabell som
                                      ; holder pekere til alle
                                      ; navnopbjekter indeksert på
                                      ; navnet

(setq *forrige-liste* (vector '"" '"" '"" '"" '"" '"" '""))
; Vektor med siste navneform, brukes til å skifte ut &mdash; med
; korrekt form.

(setq *tittel-kategori* (make-hash-table :test #'equal))
; Hashtabell som holder kobling tittel-kategori

(setq *navn-navneformer* (make-hash-table :test #'equal))
; Hashtabell som holder kobling navn-liste over navneformer

(setq *alle-oppslags-pnavn* '()) ; Liste med alle
                                 ; oppslags-pnavn. Settes av
                                 ; sett-alle-oppslags-pnavn som kjøres
                                 ; av sett-inn-pnavn-oppslag-noder.

(setq *sidetall-nodeid* (make-hash-table)) ; Hashtabell som holder
                                           ; alle sidetall med et par
                                           ; som er id'ene til første
                                           ; og siste node på sida.
```

```
(setq *foresporsler* 0) ; Antall forspørsler til bruker.

(setq *mulig-talerliste* '()) ; Objektene til de navn som ble
                                ; nevnt i forrige avsnitt. Forrige
                                ; taler pushes på.

(setq *uspesifisert-skriver* '()) ; Dummy-pnavn for den som fører
                                ; pennen.

(setq *skal-ikke-analyseres* '()) ; Dummy-pnavn for avsnitt som er for
                                ; korte eller blandede for analyse.


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Klassedeklarasjon for navnestruktur                                   ;
;-----------------------------------------------------------------------;

(defclass navn ()
  ((id :initarg :id :reader navnid)
   (navn :initarg :navnet :reader navnet)
   (navneformer :initform '() :accessor navneformer)
   (reg-elementliste :initform '() :accessor navnets-reg-elementer)
   (elementliste :initform '() :accessor navnets-elementer)
   (sidehenv :initform '() :accessor navnets-sidehenv)
   (se-henv :initform '() :accessor navnets-sehenv))
  (:documentation "Hovedklasse for navn og inneholder id, navnet,
navneformer, lister over SGML-elementer for innførsler i register og
for andre forekomster, sidehenvisninger og se-henvisninger.
Navneformer er ei liste av lister. Se-henvisninger er ei liste av par:
navneform som henvises til og id for noden det henvises til."))

(defclass pnavn (navn)
  ((kategori :initform "" :accessor pnavnkategori)
   (tittel :initform "" :accessor pnavntittel)
   (taleravsnitt :initform '() :accessor pnavntaleravsn))
  (:documentation "Klasse for personnavn."))

(defclass snavn (navn)
  ((koordinater :initform "" :accessor stedskoordinater))
  (:documentation "Klasse for stedsnavn."))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; SGML-tre-manipulasjon: legg inn sidetall                              ;
;-----------------------------------------------------------------------;

(defun sett-inn-sidetall-paa-alle-noder ()
  "Legger inn sidetall på alle noder basert på sidetallene i snr-node
  i samme side-intervall, eventuelt basert på regneregler."
  (let ((sidenoder (finn-node-ider '"SIDE"))
        (sidetallnoder (finn-node-ider '"SNR")))
;;; Begynner med å legge inn sidetallet på alle SNR-noder:
    (legg-inn-sidetall-paa-snr sidetallnoder)
;;; Fortsetter med å legge inn de andre sidetallene
    (legg-inn-sidetall-foer-foerste-side sidenoder sidetallnoder)
    (legg-inn-sidetall sidenoder sidetallnoder))
```

```
  (format t "~%Lagt inn sidetall på nodene..."))

(defun legg-inn-sidetall-paa-snr (sidetallnoder)
  "Legger inn sidetall på alle SNR-noder."
  (mapc (lambda (sidetallnodeid)
          (let* ((sidetallnode (finn-node sidetallnodeid))
                 (sidetall (finn-attributtverdi
                            (elementattributtliste sidetallnode) '"NR")))
            (if sidetall
                (progn
                  (if (romanp sidetall)
                      (progn
                        (setf (sidetall sidetallnode) (roman2arabic sidetall))
                        (setf (sidetalltype sidetallnode) '"romer"))
                    (progn
                      (setf (sidetall sidetallnode) (parse-integer sidetall))
                      (setf (sidetalltype sidetallnode) '"arabisk"))))
              (error "sett-inn-sidetall-på-noder: Fant ikke sidentall på snr-node: "
                     sidetallnodeid))))
        sidetallnoder))

(defun legg-inn-sidetall-foer-foerste-side (sider sidetall)
  "Setter inn sidetall før første side-node."
  (when (car sider)
    (let* ((forste-side (car sider))
           (forste-sidetall (car sidetall))
           (sidetallnode (finn-node forste-sidetall))
           (sidetall (sidetall sidetallnode))
           (sidetalltype (sidetalltype sidetallnode))
           (pos-i-sider (position forste-sidetall
                                  sider
                                  :test #'(lambda (x y) (< x y)))))
      (sett-inn-sidetall-paa-nodene 1 forste-side (- sidetall pos-i-sider)
                                    sidetalltype))))

(defun legg-inn-sidetall (sider sidetall)
  "Setter inn sidetall på alle noder etter følgende regler: 1. Dersom
   det finnes en snr mellom to side settes alle noder mellom de to side
   til nr i snr.  2. Etter siste side settes til verdier i snrs nr etter
   siste side."
  (when (car sider)
    (let* ((side1 (car sider))
           (side2 (if (cadr sider)
                      (cadr sider)
                    (+ *idnummer* 1))) ; Etter siste sidetagg.
             ;;; Her kan det hende *idnummer* er lik side2, slik at
             ;;; det ikke er mer å gjøre, men det skader ikke.
           (mulig-snr (filter #'(lambda (x) (when (and (> x side1)
                                                       (< x side2))
                                              x))
                              sidetall)))
      (cond ((not mulig-snr)
             (legg-inn-sidetall (cdr sider) sidetall))
            ;;; Må finne på noe lurt her, før legg-inn-sidetall kalles på nytt.
            ((= (length mulig-snr) 1)
             (let* ((sidetallnode (finn-node (car mulig-snr)))
                    (sidetallet (sidetall sidetallnode))
```

```
                (sidetalltypen (sidetalltype sidetallnode)))
            (sett-inn-sidetall-paa-nodene side1 side2 sidetallet sidetalltypen)
            (legg-inn-sidetall (cdr sider) sidetall)))
          (t ; Flere snr-tagger mellom to side-tagger. Semantisk feil.
           (error "legg-inn-sidetall: Flere enn en snr-tag mellom to side-tagger: "
                  mulig-snr))))))

(defun sett-inn-sidetall-paa-nodene (fra til sidetall sidetalltype)
  "Setter inn sidetall på nodene i spennet."
  (setf (gethash sidetall *sidetall-nodeid*) (cons (+ fra 1) (- til 1)))
  (mapc (lambda (nodeid)
          (let ((node (finn-node nodeid)))
            (setf (sidetall node) sidetall)
            (setf (sidetalltype node) sidetalltype)))
        (enumerate-interval (+ fra 1) (- til 1))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Funksjoner for navnestruktur                                          ;
;-----------------------------------------------------------------------;

(defun legg-til-i-pnavnoppsl (navn navn-node)
  "Legger til i *pnavnoppsl*. Nodene ligger i ei liste slik at det
blir plass til flere på samme navneform."
  (let ((navn-hash-oppsl (gethash navn *pnavnoppsl*)))
    (setf (gethash navn *pnavnoppsl*)
          (if navn-hash-oppsl
              (list navn-node (cdr navn-hash-oppsl))
            (list navn-node)))))

(defun lag-pnavn (pnavn)
  "Lager et pnavn-element og returnerer id'en det får."
  (let* ((id (incf *navnnummer*))
         (nytt-pnavn (make-instance 'pnavn :navnet pnavn :id id)))
    (setf (gethash id *navnene*) nytt-pnavn)
    (legg-til-i-pnavnoppsl (rydd-streng pnavn) nytt-pnavn)
    id))

(defun finn-navn (id)
  "Returnerer peker til objektet med gitt id"
  (gethash id *navnene*))

(defun navn-nytt-reg-element (navn node)
  "Legger et nytt element inn i registerlista hvis det ikke finnes fra før."
  (setf (navnets-reg-elementer navn)
        (remove-duplicates (concatenate 'cons (navnets-reg-elementer navn)
                                        (list node)))))

(defun navn-nytt-element (navn node)
  "Legger et nytt element inn i nodelista hvis det ikke finnes fra før."
  (setf (navnets-elementer navn)
        (remove-duplicates (concatenate 'cons (navnets-elementer navn)
                                        (list node)))))

(defun navn-ny-sidehenv (navn sidetalliste)
  "Legger ett eller flere nye elementer inn i sidehenvlista hvis det
   ikke finnes fra før."
```

```
  (setf (navnets-sidehenv navn)
        (remove-duplicates (concatenate 'cons (navnets-sidehenv navn)
                                        sidetalliste))))

(defun pnavn-sett-kategori (pnavn kat)
  "Setter ny kategori på navneobjektet."
  (if (typep pnavn 'pnavn)
      (setf (pnavnkategori pnavn) kat)
    (error "pnavn-sett-kategori: ikke pnavn")))

(defun pnavn-sett-tittel (pnavn tit)
  "Setter ny tittel på navneobjektet."
  (if (typep pnavn 'pnavn)
      (setf (pnavntittel pnavn) tit)
    (error "pnavn-sett-tittel: ikke pnavn")))

(defun pnavn-nytt-taleravsn-element (navn node)
  "Legger et nytt element inn i taleravsn hvis det ikke finnes fra før."
  (setf (pnavntaleravsn navn)
        (remove-duplicates (concatenate 'cons (pnavntaleravsn navn)
                                        (list node)))))

(defun navn-sett-sehenv (navn sehenv-par)
  "Setter inn sehenv-paret i se-henvisningslista i navneobjektet."
  (setf (navnets-sehenv navn) (cons sehenv-par (navnets-sehenv navn))))


(defun slaa-sammen-med-parantes (navneliste &optional (status 'ikke-par)
                                                (utliste '()) (utstreng ""))
  "Slår sammen variasjoner med navnet foran i lista."
  (if (null navneliste)
      (reverse utliste)
    (let* ((navnedel (car navneliste))
           (navnedel-siste (subseq navnedel
                                   (- (length navnedel) 1)))
           (navnedel-nestsiste (if (> (length navnedel) 1)
                                   (subseq navnedel
                                           (- (length navnedel) 2)
                                           (- (length navnedel) 1))
                                 '" "))) ; Blank slår ikke ut i testene.
      (if (null (cdr navneliste))
          (if (eq status 'par)
              (reverse (push (concatenate 'string utstreng '" " navnedel)
                             utliste))
            (reverse (push navnedel utliste)))
        (let* ((neste-navnedel (cadr navneliste))
               (neste-navnedel-forste (subseq neste-navnedel 0 1))
               (neste-navnedel-siste (subseq neste-navnedel
                                   (- (length neste-navnedel) 1)))
               (neste-navnedel-nestsiste (if (> (length neste-navnedel) 1)
                                             (subseq neste-navnedel
                                                     (- (length neste-navnedel) 2)
                                                     (- (length neste-navnedel) 1))
                                           '" "))) ; Blank slår ikke ut i testene.
          (if (eq status 'par)
                                   ; Inne i parantes
              (if (or (equal navnedel-siste '")")
                      (and (equal navnedel-nestsiste '")")
```

```
                        (equal navnedel-siste '",")))
                                    ; Paranteses avsluttes
                  (slaa-sammen-med-parantes (cdr navneliste)
                                            'ikke-par
                                            (push (concatenate 'string
                                                               utstreng
                                                               '" "
                                                               navnedel)
                                                  utliste)
                                            '"")
                                    ; Parantesen fortsetter
                     (slaa-sammen-med-parantes (cdr navneliste)
                                               status
                                               utliste
                                               (concatenate 'string
                                                            utstreng
                                                            '" "
                                                            navnedel)))
                                    ; Ikke i parantes
            (if (and (not (equal navnedel-siste '","))
                     (equal neste-navnedel-forste '"("))
                                    ; Parantes starter
                (if (or (equal neste-navnedel-siste '")")
                        (and (equal neste-navnedel-nestsiste '")")
                             (equal neste-navnedel-siste '",")))
                                    ; Parantes slutter med en gang
                   (slaa-sammen-med-parantes (cddr navneliste)
                                             status
                                             (push (concatenate 'string
                                                                navnedel
                                                                '" "
                                                                neste-navnedel)
                                                   utliste)
                                             '"")
                                    ; Paranteser fortsetter
                  (slaa-sammen-med-parantes (cdr navneliste)
                                            'par
                                            utliste
                                            navnedel))
                                    ; Ingen parantes i det hele tatt
              (slaa-sammen-med-parantes (cdr navneliste)
                                        status
                                        (push navnedel utliste)
                                        '"")))))))


(defun kompletter-pnavn (navneform)
  "Dersom navneform inneholder &mdash; legger tidligere verdi inn på
   gitt posisjon."
  (let ((navneliste (slaa-sammen-med-parantes ; Tar hensyn til
                                              ; variasjoner i parantes
                     (split navneform (code-char 32))))
        (navn-ut '()))
;    (format t "~%Navneliste: ~{<~A> ~}" navneliste)
    (do* ((nr 0 (+ nr 1))
          (forrige (svref *forrige-liste* nr) (svref *forrige-liste* nr))
          (navn navneliste (cdr navn)))
         ((null navn) (lag-streng-av-liste navn-ut))
```

```
        (cond ((equal (car navn) '"&mdash;") ; Gjentakelse av fornavn
               (setq navn-ut (append navn-ut (list forrige))))
              ((equal (car navn) '"&mdash;,") ; Gjentakelse av
                                              ; foranstilt etternavn
               (setq navn-ut (append navn-ut
                                     (list
                                      (concatenate
                                       'string
                                       forrige
                                       (unless (equal (siste-tegn forrige) #\,)
                                         '",")))))))
              (t
               (setq navn-ut (append navn-ut (list (car navn))))
               (setf (svref *forrige-liste* nr) (car navn)))))))

(defun plukk-henv-sidetall (nodeliste)
  "Returnerer ei liste bestående av alle sidetall i nodene i
   nodelista. Sidetallene leveres som strenger, pga. 'f.' o.l.."
  (if (null nodeliste)
      '()
    (flat-ut (mapcar #'(lambda (x) (split x '". "))
                     (split (finn-pcdata-fra-tre nodeliste) #\,)))))

(defun se-henvisningp (oppslnode)
  "Sjekker om en oppslagsnode representerer en se-henvisning. Rått
   tilhogd, men funker. Hvis ja: returnerer pnavn-noden etter 'se'."
  (let ((tekstnode (finn-pcdatanode-med-txt (list oppslnode) '", se ")))
    (when (and (> (length (finn-noder-i-subtre (list oppslnode) 'pnavn))
                  1)
               tekstnode)
      (let ((returliste (filter (lambda (node)
                                  (when (and (typep node 'element)
                                             (equal (elementnavn node) '"pnavn"))
                                    node))
                                (sosken-etter tekstnode))))
        (if returliste
            (car returliste)
          (error
           "se-henvisningp: Finner ikke pnavn som henviningen skal henvise til!"))))))

(defun lag-navnereg-innforsel (oppslnode)
  "Lager et navneobjekt med registerinformasjon basert på subtreet
   under oppslnode og legger dette inn."
  (let ((navnenoder (finn-noder-i-subtre (list oppslnode) '"pnavn")))
    (if navnenoder
        (let* ((oppslagsnavnenode (car navnenoder))
               (innforsel-navneforslag
                (kompletter-pnavn
                 (rydd-streng (finn-pcdata-fra-tre (list oppslagsnavnenode))))))
          (multiple-value-bind
           (innforsel-navn aarsak)
           (if (find #\, innforsel-navneforslag)
               (spor-bruker-etter-oppslagsform-pnavn innforsel-navneforslag
                                                     '"Komma i navnet.")
             (values innforsel-navneforslag '""))
           (let* ((pnavn-id (lag-pnavn innforsel-navn))
                  (navnet (finn-navn pnavn-id))
```

```
                (sehenv (se-henvisningp oppslnode))
                (tittel (if sehenv
                            '""
                          (finn-tittel oppslnode pnavn-id)))
                (kategori (gethash tittel *tittel-kategori*)))
;;; BEGRUNNELSE:
            (ny-grunn
             (incf *grunnid*)
             '"Innførsel i navneregisteret"
             (concatenate
              'string
              '"Første pnavn-node (objekt 1) i oppslagsnoden (objekt 2) lagt inn
som objekt i navneregisteret (objekt 3)."
                                  (unless (equal aarsak '"")
                                    (concatenate 'string '"
Fra bruker: " aarsak)))
                      '"parse-dokub: lag-navnereg-innforsel"
                      (list (cons 'node (nodeid oppslagsnavnenode))
                            (cons 'node (nodeid oppslnode))
                            (cons 'navn pnavn-id)))
            (navn-nytt-reg-element navnet oppslnode)
            (navn-ny-sidehenv
             navnet
             (plukk-henv-sidetall (finn-noder-i-subtre (list oppslnode) 'henv)))
            (when sehenv
              (navn-sett-sehenv
               navnet
               (cons (rydd-streng (finn-pcdata-fra-tre (list sehenv))) '())))
            (pnavn-sett-tittel navnet tittel)
            (pnavn-sett-kategori navnet kategori)
            pnavn-id)))
      (error "lag-navnereg-innforsel: finnes ikke navnenoder i subtreet.")))

(defun lag-navnereg ()
  "Lager et register av alle navnenoder basert på alle
   oppslagsnavn. Først i registeret et dummy skriver-navn."
  (let ((pnavn-id (lag-pnavn '"Uspesifisert skriver")))
    (setq *uspesifisert-skriver* (finn-navn pnavn-id))
;;; BEGRUNNELSE:
    (ny-grunn (incf *grunnid*)
              '"Innførsel i navneregisteret"
              '"Dummy-innførsel."
              '"parse-dokub: lag-navnereg"
              (list (cons 'navn pnavn-id)))
    (pnavn-sett-tittel *uspesifisert-skriver* '"skriver")
    (pnavn-sett-kategori *uspesifisert-skriver* '"embedsmann"))
  (let* ((pnavn-id (lag-pnavn '"Forhører"))
         (pnavn (finn-navn pnavn-id)))
;;; BEGRUNNELSE:
    (ny-grunn (incf *grunnid*)
              '"Innførsel i navneregisteret"
              '"Dummy-innførsel."
              '"parse-dokub: lag-navnereg"
              (list (cons 'navn pnavn-id)))
    (pnavn-sett-tittel pnavn '"forhører")
    (pnavn-sett-kategori pnavn '"embedsmann"))
  (let ((pnavn-id (lag-pnavn '"Skal ikke analyseres")))
```

```
    (setq *skal-ikke-analyseres* (finn-navn pnavn-id))
;;; BEGRUNNELSE:
    (ny-grunn (incf *grunnid*)
              '"Innførsel i navneregisteret"
              '"Dummy-innførsel."
              '"parse-dokub: lag-navnereg"
              (list (cons 'navn pnavn-id))))
  (mapc #'lag-navnereg-innforsel (mapcar #'finn-node (finn-node-ider 'pnavnoppsl)))
  (format t "~%Laget navneregister..."))


(defun pnavn-oppslagp (node)
  "Representerer denne noden et oppslagsnavn i registeret?"
  (if (and (typep node 'element)
           (equal (string-downcase (elementnavn node)) '"pnavn")
           (or (finnes-element-i-subtre (nodebarn node) '"kur")
               (finnes-tekst-i-subtre (nodebarn node) '"&mdash;")))
      node))


(defun sett-alle-oppslags-pnavn ()
  "Lister opp alle nodene som representerer pnavn-oppslag."
  (setq *alle-oppslags-pnavn*
        (filter (lambda (x)
                  (pnavn-oppslagp (finn-node x)))
                (filter (lambda (x)
                          (when (and (> x *pnavnreg-idstart*)
                                     (< x *pnavnreg-idslutt*))
                            x))
                        (finn-node-ider '"pnavn")))))

(defun neste-oppslags-node (node)
  "Finner neste node som representerer pnavn-oppslag. Returnerer nil hvis ikke fler."
  (let ((pos (position node *alle-oppslags-pnavn*)))
    (if pos
        (if (= (length *alle-oppslags-pnavn*)
               (+ pos 1))
            '() ; node er siste pnavn-oppslagsnode
          (elt *alle-oppslags-pnavn* (+ pos 1)))
      (error (concatenate 'string
                          "neste-oppslags-node: node er ikke en oppslagsnode"
                          (nodeid node))))))

(defun sett-inn-pnavn-oppslag-noder ()
  "Personnavn-oppslagnoder skal settes inn slik at de lager et subtre
   av alt som representerer informasjon om et pnavn i registeret. Dette
   er: Etterfølgende søsken fram til neste pnavn-oppslagsnode.
   Setter den globale lista over oppslagsnoder først."
  (sett-alle-oppslags-pnavn)
  (mapc #'sett-inn-pnavn-oppslag-node *alle-oppslags-pnavn*)
  (format t "~%Satt inn oppslagselementene i treet..."))


(defun sett-inn-pnavn-oppslag-node (node)
  "Setter inn en oppslagsnode over noden gitt reglene i
sett-inn-pnavn-oppslag-noder."
  (let* ((mamma (nodens-mamma node))
         (ny-node (lag-element '"pnavnoppsl" '() mamma (incf *idnummer*)))
```

```
         (neste-oppslags-node (neste-oppslags-node node))
                                       ; Går ut fra at det er dette
                                       ; kallet som tar så lang tid.
         (neste-node (if (and neste-oppslags-node
                              (eq (nodens-mamma node)
                                       ; Hvis neste oppslagsnode er i
                                       ; samme søskenflokk, skal
                                       ; oppslaget stoppes før den.
                                  (nodens-mamma neste-oppslags-node)))
                         neste-oppslags-node
                       '())))
    (sett-inn-node ny-node mamma node neste-node)))

(defun finn-pnavn-oppslag (navn)
  "Finner pnavn-oppslag som stemmer med navnet."
  (gethash navn *pnavnoppsl*))

(defun navnets-sehenv-navn (navn)
  "Returnerer ei liste med navnedelene av sehenv-parene."
  (mapcar #'car (navnets-sehenv navn)))

(defun finn-tittel (navneoppsl pnavn-id)
  "Regner seg fram til tittelen på personen.
   Regel 1: Første ord etter pnavn-noden.
   Regel 2: Sjekker om første fire ord etter pnavn-noden er i registeret -
            ofte er tittelen fjerde ord.
   Regel 3: Hvis ikke der: Spør brukeren.
   Tar med seg pnavn-id for å sende videre til grunn-registeret."
  (let ((pnavn-noder (finn-noder-i-subtre (list navneoppsl) '"pnavn")))
    (if pnavn-noder
        (let ((neste-node (sosken-etter (car pnavn-noder))))
          (if (and neste-node
                   (typep (car neste-node) 'pcdata))
              (let ((ordene (hent-ord (rydd-streng (finn-pcdata-fra-tre neste-node))
                                      '".")))
                (if ordene
                    (let* ((tittel1 (car ordene))
                           (tittel2 (if (> (length ordene) 1)
                                        (cadr ordene)
                                      tittel1))
                           (tittel3 (if (> (length ordene) 2)
                                        (caddr ordene)
                                      tittel2))
                           (tittel4 (if (> (length ordene) 3)
                                        (cadddr ordene)
                                      tittel3)))
                      (cond
                       ((gethash tittel1 *tittel-kategori*)
;;; BEGRUNNELSE:
                        (ny-grunn
                         (incf *grunnid*)
                         '"Innførsel i tittelregister"
                         '"Fant igjen første ord etter pnavn i tittel-kategoriregister."
                         '"parse-dokub: finn-tittel"
                         (list (cons 'navn pnavn-id)
                               (cons 'node (nodeid navneoppsl))))
                        tittel1)
```

```
                      ((gethash tittel2 *tittel-kategori*)
;;; BEGRUNNELSE:
                       (ny-grunn
                        (incf *grunnid*)
                        '"Innførsel i tittelregister"
                        '"Fant igjen andre ord etter pnavn i tittel-kategoriregister."
                        '"parse-dokub: finn-tittel"
                        (list (cons 'navn pnavn-id)
                              (cons 'node (nodeid navneoppsl))))
                       tittel2)
                      ((gethash tittel3 *tittel-kategori*)
;;; BEGRUNNELSE:
                       (ny-grunn
                        (incf *grunnid*)
                        '"Innførsel i tittelregister"
                        '"Fant igjen tredje ord etter pnavn i tittel-kategoriregister."
                        '"parse-dokub: finn-tittel"
                        (list (cons 'navn pnavn-id)
                              (cons 'node (nodeid navneoppsl))))
                       tittel3)
                      ((gethash tittel4 *tittel-kategori*)
;;; BEGRUNNELSE:
                       (ny-grunn
                        (incf *grunnid*)
                        '"Innførsel i tittelregister"
                        '"Fant igjen fjerde ord etter pnavn i tittel-kategoriregister."
                        '"parse-dokub: finn-tittel"
                        (list (cons 'navn pnavn-id)
                              (cons 'node (nodeid navneoppsl))))
                       tittel4)
                      (t
                       (multiple-value-bind
                           (tittel aarsak)
                           (spor-bruker-etter-tittel
                            ordene
                            navneoppsl
                            '"Finner ikke tittelen i tittellista.")
;;; BEGRUNNELSE:
                         (ny-grunn (incf *grunnid*)
                                   '"Innførsel i tittelregister"
                                   (concatenate 'string
                                                '"Bruker har valgt tittel."
                                                (unless (equal aarsak '"")
                                                  (concatenate 'string '"
Fra bruker: " aarsak)))
                                   '"parse-dokub: finn-tittel"
                                   (list (cons 'navn pnavn-id)
                                         (cons 'node (nodeid navneoppsl))))
                           tittel))))
                  (multiple-value-bind
                      (tittel aarsak)
                      (spor-bruker-etter-tittel '""
                                                navneoppsl
                                                '"Finner ingen tittel i oppslaget.")
;;; BEGRUNNELSE:
                    (ny-grunn (incf *grunnid*)
                              '"Innførsel i tittelregister"
```

```
                              (concatenate 'string
                                           '"Bruker har valgt tittel."
                                           (unless (equal aarsak '"")
                                             (concatenate 'string '"
Fra bruker: " aarsak)))
                              '"parse-dokub: finn-tittel"
                              (list (cons 'navn pnavn-id)
                                    (cons 'node (nodeid navneoppsl))))
                     tittel)))
            (error "finn-tittel: Ingen pcdata etter pnavn-noden.")))
      (error "finn-tittel: Ingen pnavn-node.")))

(defun koble-alle-pnavnreg-til-pnavn ()
  "Legger inn aktuelle noder i teksten i navnets-elementer på alle
   pnavn-objekter."
  (maphash #'(lambda (x y)
               (koble-ett-pnavnreg-til-pnavn y))
           *navnene*)
  (format t "~%Satt inn pekere fra pnavnreg til pnavnnoder i teksten..."))

(defun finn-noder-paa-side (navn side)
  "Finner alle nodene som representerer et element på side, evt. fram
   til side-til."
  (let ((nodeidnrene (gethash side *sidetall-nodeid*)))
    (when nodeidnrene
      (mapcar #'finn-node (finn-node-ider navn (car nodeidnrene)
                                         (cdr nodeidnrene))))))

(defun koble-ett-pnavnreg-til-pnavn (pnavn)
  "Legger inn aktulle noder i teksten i navnets-elementer på pnavn."
  (let ((sidene (navnets-sidehenv pnavn)))
    (if sidene
        (let ((enkeltsider (plukk-sider sidene)))
          (mapc #'legg-pnavn-paa-pnavnreg
                (make-list (list-length enkeltsider)
                           :initial-element pnavn)
                enkeltsider)))))


(defun plukk-sider (sidestrenger)
  "Plukker ut enkeltsidetall fra sidestrenger. f. tolkes som neste
   side, ff. som de fire neste sidene. Henvisninger til andre bind fjernes."
  (if (null sidestrenger)
      '()
    (let ((sidestreng (car sidestrenger)))
      (if (search '"(II)" sidestreng)
          (plukk-sider (cdr sidestrenger))
        (let ((bindestrek (search '"-" sidestreng)))
          (if bindestrek
              (append (enumerate-interval
                       (parse-integer (subseq sidestreng 0 bindestrek))
                       (parse-integer (subseq sidestreng (+ bindestrek 1))))
                      (plukk-sider (cdr sidestrenger)))
            (let ((side (parse-integer sidestreng :junk-allowed t))
                  (ff (search '"ff" sidestreng)))
              (if ff
                  (append (enumerate-interval side (+ side 4))
                          (plukk-sider (cdr sidestrenger)))
```

```
               (let ((f (search '"f" sidestreng)))
                 (if f
                     (cons side (cons (+ side 1) (plukk-sider (cdr sidestrenger))))
                   (cons side (plukk-sider (cdr sidestrenger))))))))))))


(defun legg-pnavn-paa-pnavnreg (pnavn side)
  "Finner ett eller flere eksemplarer av pnavn på side i teksten og
   legger peker til nodene i navnets-elementer på pnavn."
  (let* ((noder-paa-sida (finn-noder-paa-side '"pnavn" side))
         (navn-paa-sida (mapcar #'(lambda (node)
                                    (rydd-streng (finn-pcdata-fra-tre (list node))))
                                noder-paa-sida)))
    (do ((nodene noder-paa-sida (cdr nodene))
         (navnene navn-paa-sida (cdr navnene)))
        ((null nodene) t)
        (when (sammenlign-stedsnavn (car navnene) pnavn)
          (setf (registerobjekt (car nodene)) pnavn)
          (navn-nytt-element pnavn (car nodene))
;;; BEGRUNNELSE:
          (ny-grunn (incf *grunnid*)
                    '"Kobling pnavn-node."
                    '"Soundex-kodingen av innholdet i noden (objekt 1)
er gjenfunnet i soundex-koding av en av navneformene i
pnavn-objektet (objekt 2). Sidene man leter på er hentet
fra sideinnførsler på objekt 2 etter vanlige regler."
                    '"parse-dokub: legg-pnavn-paa-pnavnreg"
                    (list (cons 'node (nodeid (car nodene)))
                          (cons 'navn (navnid pnavn))))))))

(defun sammenlign-stedsnavn (navn oppslnavnobj)
  "Sammenligner for å se om de kan representere samme navn ved å
   sjekke lista navneformer i objektet. Sikrer mot tomme strenger i navneformlista."
  (finn-igjen-soundex navn (filter #'(lambda (x) (when (not (tom-strengp x)) x))
                                   (navneformer oppslnavnobj))))

(defun legg-inn-navneformer-paa-alle ()
  "Legger inn navneformene på alle navn."
  (maphash #'(lambda (x y)
               (legg-inn-navneformer-paa-ett y))
           *navnene*)
  (format t "~%Satt inn navneformene på alle navn..."))

(defun legg-inn-navneformer-paa-ett (navn)
  "Legger inn navneformene på et navn."
  (let* ((navnet (navnet navn))
         (former-oppslag (gethash navnet *navn-navneformer*)))
    (cond (former-oppslag
           (setf (navneformer navn) former-oppslag)
;;; BEGRUNNELSE:
           (ny-grunn (incf *grunnid*)
                     '"Navneformer på et navneobjekt"
                     '"Navnet (objekt 1) finnes i *navn-navneformer*,
så navneformene legges inn derfra."
                     '"parse-dokub: legg-inn-navneformer-paa-ett"
                     (list (cons 'navn (navnid navn))))
           former-oppslag)
          ((search '"(" navnet)
```

```
          (multiple-value-bind
              (navneformer aarsak)
              (spor-bruker-etter-parformer
               navnet
               (append
                (list
                 (string-trim
                  '" "
                  (rydd-streng
                   (byttut '"" '")"
                           (byttut '"" '"(" navnet)))))
                (list (string-trim '" "
                                   (rydd-streng (fjern-paranteser navnet))))
                (list (string-trim '" "
                                   (rydd-streng (fjern-paranteser navnet '" "))))
                (erstatt-parantestekst navnet)) '"Navnet inneholder paranteser.")
;;; BEGRUNNELSE:
            (ny-grunn
             (incf *grunnid*)
             '"Navneformer på et navneobjekt"
             (concatenate 'string
                          '"Navnet (objekt 1) inneholder parantes, så bruker ble spurt:
"
                          aarsak)
             '"parse-dokub: legg-inn-navneformer-paa-ett"
             (list (cons 'navn (navnid navn))))
            (setf (navneformer navn) navneformer)
            (setf (gethash navnet *navn-navneformer*) navneformer)
            navneformer))
          (t
           (multiple-value-bind
               (navneformer aarsak)
               (snu-navn navnet)
             (setf (navneformer navn) navneformer)
             (setf (gethash navnet *navn-navneformer*) navneformer)
;;; BEGRUNNELSE:
             (ny-grunn (incf *grunnid*)
                       '"Navneformer på et navneobjekt"
                       (concatenate 'string
                                    '"Navnet (objekt 1) får navneformer.
"
                                    aarsak)
                      '"parse-dokub: legg-inn-navneformer-paa-ett"
                      (list (cons 'navn (navnid navn))))
             navneformer)))))

(defun snu-navn (navnet)
  "Snur navnet dersom det er på formen ord, ord+"
  (let ((komma (search '"," navnet)))
    (if komma
        (multiple-value-bind
         (parformer aarsak)
         (spor-bruker-etter-parformer
          navnet
          (list
           (string-trim
            '" "
```

```
            (concatenate
             'string
             (subseq navnet
                     (+ komma 1))
             '" "
             (subseq navnet
                     0
                     komma)))) '"Inneholder komma")
        (values parformer
                (concatenate
                 'string
                 '"Navnet (objekt 1) inneholder komma, så bruker ble spurt:
"
                 aarsak)))
      (values (list navnet) '"Bruker ble ikke spurt - forslaget var uproblematisk.")))

(defun finn-pnavn-uten-pnavnreg ()
  "Finner alle pnavn-elementer som ikke har koblet til seg et navnobjekt."
  (mapcar #'(lambda (x) (when (not (registerobjekt (finn-node x)))
                          (finn-pcdata-fra-tre (list (finn-node x)))))
          (finn-node-ider '"pnavn" 2 *pnavnreg-idstart*)))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Navnestruktur: Paranteser                                    ;
;--------------------------------------------------------------;

(defun fjern-paranteser (streng &optional (erstattes-med '""))
  "Fjerner paranteser med innhold fra streng og setter inn
   erstattes-med i stedet."
  (let ((par1 (search '"(" streng))
        (par2 (search '")" streng)))
    (if (and par1 par2)
        (fjern-flere-blanke
         (concatenate 'string (subseq streng 0 par1)
                      erstattes-med
                      (fjern-paranteser (subseq streng (+ par2 1)) erstattes-med)))
      streng)))

(defun erstatt-parantestekst (streng &optional (start1 0) (start2 0))
  "Returnerer ei liste med hvert element i parantes i stedet for ordet foran."
  (if (> (length streng) start2)
      (let ((par1 (search '"(" streng :start2 start1))
            (par2 (search '")" streng :start2 start2)))
        (if (and par1 par2 (> par1 0)) ; Sjekker også at ikke navnet
                                       ; innledes med parantes
            (let ((foer-paranteslista (hent-ord (subseq streng 0 par1) '"()"))
                  (paranteslista (hent-ord (subseq streng (+ par1 1) par2)))
                  (etter-parantes (fjern-paranteser (subseq streng (+ par2 1))))
                  (utliste '()))
              (dolist
               (parantesord paranteslista)
               (setf utliste
                     (cons
                      (string-trim
                       '" "
                       (rydd-streng
```

```
                      (concatenate
                       'string
                       (fjern-paranteser
                        (lag-streng-av-liste
                         (subseq
                          foer-paranteslista
                          0
                          (- (length foer-paranteslista) 1))))
                       '" "
                       parantesord
                       etter-parantes)))
                    utliste)))
            (append (reverse utliste)
                    (erstatt-parantestekst streng (+ par1 1) (+ par2 2))))
          (list streng)))
    (list streng)))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Navnestruktur: Interaksjon                                           ;
;----------------------------------------------------------------------;

(defun begrunnelse-fra-bruker ()
  "Ber bruker om en begrunnelse. Spesifiserer ikke årsak."
  (format t "~%Hvorfor det? ")
  (read-line))

(defun skriv-alternativer (alt sporrtxt)
  "Lister opp alternativene med nummer."
  (let ((teller 0))
    (dolist (txt alt)
      (format t "~%Alternativ ~D: <~A>"
              (incf teller)
              txt))
    (format t "~%~%~A" sporrtxt)))

;;; Spør etter tittel

(defun spor-bruker-etter-tittel (forslag navnoppsl aarsak)
  "Spør brukeren om det som finnes i forslag er OK som tittel til
   navnoppsl. Hvis ikke, skriv inn nytt."
  (if *brukerdialog*
      (progn
        (incf *foresporsler*)
        (format
         t "~%~%Trenger opplysninger fra bruker: En persons tittel.~%Årsak: ~A~%"
         aarsak)
        (format t "~%Hele oppslagsteksten: ~%~%---> ~A <---~%"
                (finn-pcdata-fra-tre (list navnoppsl)))
        (values (be-om-valg (cond ((consp forslag)
                                   forslag)
                                  ((tom-strengp forslag)
                                   '())
                                  (t
                                   (list forslag))))
                (begrunnelse-fra-bruker)))
    (values '"" '"Bruker ble ikke spurt - ingen tittel lagt inn.")))
```

```
(defun be-om-valg (alternativer)
  "Returnerer et valg fra brukeren ut fra alternativer, som er ei
   liste med tekster."
  (if (null alternativer)
      (format t "~%Skriv inn ny tekst: ")
      (skriv-alternativer
       alternativer
       '"Skriv inn alternativ nummer eller ny tekst hvis ingen passer: "))
  (let* ((retur (read-line))
         (tall (parse-integer retur :junk-allowed t)))
    (if tall
        (if (or (> tall (length alternativer))
                (< tall 1))
            (progn
              (format t "~%~Feil tall: ~D" tall)
              (be-om-valg alternativer))
          (let ((tittel (elt alternativer (- tall 1))))
            (if (gethash tittel *tittel-kategori*)
                tittel
              (legg-inn-som-tittel tittel))))
      (if (tom-strengp retur)
          retur
        (if (gethash retur *tittel-kategori*)
            retur
          (legg-inn-som-tittel retur))))))

(defun legg-inn-som-tittel (tittel)
  "Dersom brukeren ønsker det, legges tittel inn i registeret over titler."
  (format t "~%~%Ønsker du å legg inn <~A> som ny tittel i titteloversikten?~%
Hvis ja: Velg en kategori titlen skal knyttes til fra lista,
skriv inn en annen hvis ingen passer,
eller bare <enter> hvis du ikke har noen kategori.~%
Hvis nei: Skriv inn x.~%" tittel)
  (let ((alternativer (let ((liste '()))
                        (maphash #'(lambda (x y)
                                     (push y liste))
                                 *tittel-kategori*)
                        (sort (remove-duplicates liste :test #'equal) #'string<))))
    (skriv-alternativer
     alternativer
     '"Skriv inn alternativ nummer eller ny tekst hvis ingen passer: ")
    (let* ((retur (read-line))
           (tall (parse-integer retur :junk-allowed t)))
      (if tall
          (if (or (> tall (length alternativer))
                  (< tall 1))
              (progn
                (format t "~%~Feil tall: ~D" tall)
                (legg-inn-som-tittel tittel))
            (setf (gethash tittel *tittel-kategori*) (elt alternativer (- tall 1))))
        (unless (equal (string-downcase retur) '"x")
          (setf (gethash tittel *tittel-kategori*) retur)))
      tittel)))


;;; Spør etter hovedform på navn
```

```
(defun spor-bruker-etter-oppslagsform-pnavn (navneform aarsak)
  "Spør om brukeren er fornøyd med navneformen eller om den skal endres."
  (if *brukerdialog*
      (progn
        (format
         t "~%~%En navneform til navneregisteret skal sjekkes.~%Årsak: ~A"
         aarsak)
        (format t "~%Opprinnelig navneform: ~A" navneform)
        (format
         t "~%~%Skriv ny navneform dersom du vil endre, ellers bare enter: ")
        (let ((nytt-navn (read-line)))
          (if (tom-strengp nytt-navn)
              (values navneform '"Bruker godtok forslaget uendret.")
            (values nytt-navn (begrunnelse-fra-bruker)))))
    (values navneform '"Bruker ble ikke spurt - forslagene brukt uendret.")))


;;; Spør etter parantesformer til navn

(defun spor-bruker-etter-parformer (navneform forslag aarsak)
  "Spør brukeren om det er flere navneformer som bør med."
  (if *brukerdialog*
      (progn
        (incf *foresporsler*)
        (format
         t "~%~%Trenger opplysninger fra bruker: En persons navneformer.~%Årsak: ~A"
         aarsak)
        (format t "~%Opprinnelig navneform: ~A" navneform)
        (skriv-alternativer
         forslag
         '"Skriv inn numrene på de alternativene du vil ha med skilt med blank.")
        (let ((godtatte-forslag (les-alternativer forslag)))
          (format t "~%Godtatte former:~%")
          (skriv-alternativer godtatte-forslag '"")
          (format t "Hvilke andre navneformer bør være med? Avslutt med x.~%")
          (values (append (be-om-navneformer) godtatte-forslag)
                  (begrunnelse-fra-bruker))))
    (values forslag '"Bruker ble ikke spurt - forslagene brukt uendret.")))

(defun les-alternativer (alternativer)
  "Leser inn ei rekke av alterativer."
  (format t "~%~%Ønskede alternativer: ")
  (let* ((numre (mapcar #'(lambda (nr)
                            (parse-integer nr :junk-allowed t))
                        (hent-ord (read-line) '"1234567890")))
         (alternativene '())
         (feil '())
         (lengde (length alternativer)))
    (dolist (nummer
             numre
             (if feil
                 (progn
                   (format t "~%~%Feil alterativ!")
                   (les-alternativer alternativer))
               (reverse alternativene)))
            (if (and nummer
```

```
                    (> nummer 0)
                    (<= nummer lengde))
               (push (elt alternativer (- nummer 1))
                     alternativene)
             (setq feil t)))))

(defun be-om-navneformer ()
  "Ber om navneformer som returnerer som ei liste."
  (format t "> ")
  (let ((n (read-line)))
    (if (equal (string-downcase n) '"x")
        '()
      (cons n (be-om-navneformer)))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Utskrift av navnestruktur                                            ;
;----------------------------------------------------------------------;

(defun skriv-pnavn-elementliste (liste)
  "Skriver ut ei elementliste."
  (unless (null liste)
    (format t "~%Element med id ~D: " (nodeid (car liste)))
    (skriv-subtre-som-sgml (car liste))
    (format t "~%")
    (skriv-pnavn-elementliste (cdr liste))))

(defun skriv-pnavn (pnavn)
  "Skriver ut et pnavn-objekt i fullform."
  (format t "Navn ~D: ~A"
          (navnid pnavn)
          (navnet pnavn))
  (unless (null (navnets-sehenv pnavn))
    (format t " (se~{ ~A~})"
            (navnets-sehenv-navn pnavn)))
  (format t "~%Tittel: ~A" (pnavntittel pnavn))
  (format t "~%Kategori: ~A" (pnavnkategori pnavn))
  (format t "~%Sidehenvisninger:~{ ~A;~}"
          (navnets-sidehenv pnavn))
  (format t "~%Navneformer:~{ <~A>~}" (navneformer pnavn))
  (format t "~%~%Følgende registerelementer:~%")
  (skriv-pnavn-elementliste (navnets-reg-elementer pnavn))
  (format t "~%Følgende andre elementer:~%")
  (skriv-pnavn-elementliste (navnets-elementer pnavn))
  (format t "~%Taler på følgende avsnitt: ~{ ~D;~}~%"
          (mapcar #'nodeid (pnavntaleravsn pnavn)))
  (let ((grunner (gethash (navnid pnavn) *navn-grunn*)))
    (if grunner
        (progn
          (format t "~%Begrunnelser knyttet til navnet:~%")
          (mapcar #'skriv-grunn grunner)
          (format t "~%"))
      (progn
        (format t "~%Ingen begrunnelser knyttet til navnet.")
        (format t "~%============================================================~%")))))

(defun skriv-pnavn-kort (pnavn)
```

```
  "Skriver ut et pnavn-objekt med de viktigste opplysningene."
  (format t "Navn ~D: ~A"
          (navnid pnavn)
          (navnet pnavn))
  (format t "~%Tittel: ~A.  " (pnavntittel pnavn))
  (format t "Kategori: ~A" (pnavnkategori pnavn)))

(defun skriv-pnavnene ()
  "Skriver ut alle navnene."
  (maphash (lambda (x y)
             (skriv-pnavn y))
           *navnene*))

(defun skriv-kategori (kategori)
  "Skriver ut en kategori."
  (format t"~%~%Kategori: ~A~%~%Inneholder titlene: " kategori)
  (maphash #'(lambda (tit kat)
               (when (string-equal kat kategori)
                 (format t "<~A> " tit)))
           *tittel-kategori*)
  (format t "~%========================================~%"))

(defun skriv-kategoriene ()
  "Skriver ut alle kategoriene."
  (let ((kategoriene (let ((liste '()))
                       (maphash #'(lambda (x y)
                                    (push y liste))
                                *tittel-kategori*)
                       (sort (remove-duplicates liste :test #'equal) #'string<))))
    (mapc #'skriv-kategori kategoriene))
  '())


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lagring av interaktive data                                          ;
;----------------------------------------------------------------------;

(defun lagre-kategorier-til-fil (filnavn)
  "Skriver tittel-kategori som par til fil med filnavn filnavn."
  (with-open-file (FIL filnavn :direction :output)
                  (maphash #'(lambda (tit kat)
                               (prin1 (cons tit kat) FIL))
                           *tittel-kategori*)))

(defun hent-kategorier-fra-fil (filnavn)
  "Leser tittel-kategori fra fil skrevet med
   lagre-kategorier-til-fil. Går via liste for å få snudd rekkefølgen
   riktig."
  (let ((postene '()))
    (with-open-file (FIL filnavn :direction :input)
                    (do ((post (read FIL '() 'slutt)
                               (read FIL '() 'slutt)))
                        ((eql post 'slutt))
                        (push post postene)))
    (dolist (post postene)
            (setf (gethash (car post) *tittel-kategori*) (cdr post)))))
```

```
(defun lagre-navneformer-til-fil (filnavn)
  "Skriver navn-navneformer som par til fil med filnavn filnavn."
  (with-open-file (FIL filnavn :direction :output)
                  (maphash #'(lambda (navn navneformer)
                               (prin1 (cons navn navneformer) FIL))
                           *navn-navneformer*)))

(defun hent-navneformer-fra-fil (filnavn)
  "Leser navn-navneformer fra fil skrevet med
   lagre-navneformer-til-fil. Går via liste for å få snudd rekkefølgen
   riktig."
  (let ((postene '()))
    (with-open-file (FIL filnavn :direction :input)
                    (do ((post (read FIL '() 'slutt)
                               (read FIL '() 'slutt)))
                        ((eql post 'slutt))
                        (push post postene)))
    (dolist (post postene)
            (setf (gethash (car post) *navn-navneformer*) (cdr post)))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Innsetting av taler                                                  ;
;----------------------------------------------------------------------;

(defun vis-avsnitt (&optional (rotinn '()))
  "Viser fram teksten i et subtre med rotnode rot som et avsnitt."
  (let ((rot (if rotinn
                 rotinn
               (progn
                 (format t "~%Nodenummer: ")
                 (finn-node (parse-integer (read-line) :junk-allowed t))))))
    (if (typep rot 'element)
        (finn-pcdata-fra-tre (list rot))
      (error "vis-avnitt: element."))))

(defun finn-neste-node-av-type (node elementnavnliste)
  "Finner neste node av en type i elementnavnliste."
  (if (or (null node)
          (and (typep node 'element)
               (find (string-downcase (elementnavn node))
                     (filter #'string-downcase elementnavnliste)
                     :test #'equal)))
      node
    (finn-neste-node-av-type (finn-node (+ (nodeid node) 1)) elementnavnliste)))

(defun velg-taler-til-alle-avsnitt (&optional (forrige (finn-node 1)))
  "Legger inn taler på alle avsnitt."
  (let ((neste (finn-neste-node-av-type forrige '("avsn" "overskr"))))
    (if (and neste
             (< (nodeid neste) *pnavnreg-idstart*))
        (let ((retur (velg-taler-til-avsnitt neste)))
          (if (eq retur 'slutt)
              (format t "~%~%Lagt inn taler på noen avsnitt.~%~%")
            (velg-taler-til-alle-avsnitt (finn-node (+ (nodeid neste) 1)))))
      (format t "~%~%Lagt inn taler på alle avsnitt.~%~%"))))
```

```
(defun velg-taler-til-avsnitt (rot)
  "Velger taler til avsnitt og legger det inn i noden dersom ingen
   taler finnes fra før. Kun på avsnitt lenger enn 100 tegn."
  (let ((tekst (vis-avsnitt rot)))
    (if (nodens-taler rot)
        (vis-avsnitt-med-taler rot)
      (progn
        (let ((nye (finn-noder-i-subtre (list rot) '"pnavn")))
          (setq *mulig-talerliste*
                (kort-ned-til
                 (reverse
                  (remove-duplicates
                   (reverse
                    (append (kort-ned-til *mulig-talerliste* 2)
                            nye
                            (list *uspesifisert-skriver*)
                            (list *skal-ikke-analyseres*)
                            (when (> (length *mulig-talerliste*) 2)
                              (kort-ned-til (subseq *mulig-talerliste* 2) 8))))))
                 9)))
        (if (< (length tekst) 100)
            (format t "~%---> ~A~%[Kort avsnitt; node ~D, side ~D]~%-----~%"
                    (byttut (concatenate 'string (string (code-char 10)) '"---> ")
                            (string (code-char 10)) tekst)
                    (nodeid rot) (sidetall rot))
          (progn
            (format t "~%~A~%[node ~D, side ~D]~%"
                    (vis-avsnitt rot) (nodeid rot) (sidetall rot))
            (let ((talervalg (spor-etter-taler)))
              (cond
               ((eq talervalg 'slutt)
                'slutt)
               ((typep talervalg 'pnavn)
;;; BEGRUNNELSE:
                (ny-grunn
                 (incf *grunnid*)
                 '"Taler til avsnitt"
                 (concatenate
                  'string
                  '"Taler til et avnsitt (objekt 1) satt til et navnobjekt (objekt 2)
"
                  (begrunnelse-fra-bruker))
                 '"parse-dokub: velg-taler-til-avsnitt"
                 (list (cons 'node (nodeid rot)) (cons 'navn (navnid talervalg))))
                (setf (nodens-taler rot) talervalg)
                (push talervalg *mulig-talerliste*)
                (pnavn-nytt-taleravsn-element talervalg rot))
               ((typep talervalg 'element)
                (unless (registerobjekt talervalg)
                  (format
                   t "~%Mangler registerobjekt til pnavn-node: ~D.
Vi prøver å lage kobling...~%"
                   (nodeid talervalg))
                  (let ((pnavn (sok-etter-taler)))
                    (setf (registerobjekt talervalg) pnavn)
                    (navn-nytt-element pnavn talervalg)
;;; BEGRUNNELSE:
```

```
                    (ny-grunn
                     (incf *grunnid*)
                     '"Kobling pnavn-node."
                     '"Bruker har valgt en registerinnførsel (objekt 1) til pnavn-noden
(objekt 2) fordi det manglet når taler skulle settes inn på avsnitt (objekt 3)."
                          '"parse-dokub: velg-taler-til-avsnitt"
                          (list (cons 'navn (navnid pnavn))
                                (cons 'node (nodeid talervalg))
                                (cons 'node (nodeid rot))))))
                     (setf (nodens-taler rot) (registerobjekt talervalg))
                     (push (registerobjekt talervalg) *mulig-talerliste*)
                     (pnavn-nytt-taleravsn-element (registerobjekt talervalg) rot)
;;; BEGRUNNELSE:
                     (ny-grunn (incf *grunnid*)
                               '"Taler til avnsitt"
                               (concatenate
                                'string
                                '"Taler til et avnsitt (objekt 1) satt til en node
(objekt 2) som er knyttet til et navnobjekt (objekt 3)
"
                                (begrunnelse-fra-bruker))
                               '"parse-dokub: velg-taler-til-avsnitt"
                               (list (cons 'node (nodeid rot))
                                     (cons 'node (nodeid talervalg))
                                     (cons 'navn (navnid (registerobjekt talervalg))))))
                    (t
                     (error "velg-taler-til-avnitt: Alvorlig feil!")))))))))))

(defun endre-taler-til-avsnitt (rot)
  "Sletter taleren til et avsnitt og legger inn ny. Taleren søkes opp
   fra registeret."
  (if (and (typep rot 'element)
           (equal (string-downcase (elementnavn rot)) '"avsn"))
      (progn
        (vis-avsnitt-med-taler rot)
        (format t "~%Endre taler til dette avsnittet.~%")
        (let ((talervalg (sok-etter-taler)))
          (if (and talervalg (typep talervalg 'pnavn))
              (progn
;;; BEGRUNNELSE:
                (ny-grunn
                 (incf *grunnid*)
                 '"Taler til avsnitt - endring"
                 (concatenate
                  'string
                  '"Taler til et avnsitt (objekt 1) endret til et navnobjekt (objekt 2)
"
                  (begrunnelse-fra-bruker))
                 '"parse-dokub: endre-taler-til-avsnitt"
                 (list (cons 'node (nodeid rot)) (cons 'navn (navnid talervalg))))
                (setf (nodens-taler rot) talervalg))
            (error "Intern feil i endre-taler-til-avsnitt: talervalg ikke pnavn-objekt"))))
    (format t "~%Kan ikke endre taler her, dette er ingen avsn-node!~%~%")))

(defun pnavn-nodens-navneobjekt (node)
  "Dersom noden er av typen pnavn, og dersom den har tilknyttet et
   navneobjekt, returneres dette."
```

```
  (if (and (typep node 'element)
           (equal (string-downcase (elementnavn node))
                  '"pnavn"))
      (registerobjekt node)))

(defun pnavn-nodens-kategori (node)
  "Dersom noden er av typen pnavn, og dersom den har tilknyttet et
   navneobjekt, returneres dettes kategori. Dersom ikke noe
   registerobjekt, returneres 'ikke-reg. Dersom ikke registerobjektet har
   noen kategori, returneres 'ikke-kat."
  (let ((reg-obj (pnavn-nodens-navneobjekt node)))
    (if reg-obj
        (let ((kat (pnavnkategori reg-obj)))
          (if kat
              kat
            'ikke-kat))
      'ikke-reg)))

(defun spor-etter-taler ()
  "Lister opp alternativer og ber brukeren velge, evt. søke i
   registeret."
  (skriv-alternativer
   (mapcar #'finn-navn-generell *mulig-talerliste*)
   '"Skriv nummer for forteller, blank for å søke (x avslutter innlegging): ")
  (let* ((input (read-line))
         (tall (parse-integer input :junk-allowed t))
         (ant-navn (length *mulig-talerliste*)))
    (cond ((and tall
                (> tall 0)
                (<= tall ant-navn))
           (elt *mulig-talerliste* (- tall 1)))
          ((equal '"x" (string-downcase input))
           'slutt)
          (t
           (sok-etter-taler)))))

(defun sok-etter-taler (&optional (taleravsn-merke '()))
  "Leter opp en taler i registeret. Merker poster som har taleravsn
   dersom parameter satt."
  (format t "~%Skriv inn ett eller flere navn du vil søke på: ")
  (let ((navn-sok (read-line))
        (funn-liste '())
        (merke-liste '()))
    (maphash #'(lambda (x y)
                 (when (finn-igjen navn-sok y)
                   (let ((oppsl (car (gethash x *pnavnoppsl*))))
                     (push x funn-liste)
                     (push (if (and taleravsn-merke oppsl (pnavntaleravsn oppsl))
                               taleravsn-merke
                             '"")
                           merke-liste))))
             *navn-navneformer*)
    (skriv-alternativer (mapcar #'(lambda (x y)
                                    (concatenate 'string x y))
                                funn-liste merke-liste)
                        "Hvilket navn? (blank for nytt søk) ")
    (let* ((retur (read-line))
```

```
           (tall (parse-integer retur :junk-allowed t)))
      (if (and tall
               (>= tall 1)
               (<= tall (length funn-liste)))
          (car (gethash (elt funn-liste (- tall 1)) *pnavnoppsl*))
        (sok-etter-taler taleravsn-merke)))))

(defgeneric finn-navn-generell (objekt)
  (:documentation "Finner navnet til et pnavn-objekt og teksten under
en pnavn-node. Returnerer nil ved andre nodetyper, alle andre
situasjoner gir feil."))

(defmethod finn-navn-generell ((objekt element))
  "Henter PCDATA-teksten i subtreet under noden dersom det er en pnavn-node."
  (when (equal (string-downcase (elementnavn objekt)) '"pnavn")
    (rydd-streng (finn-pcdata-fra-tre (list objekt)))))

(defmethod finn-navn-generell ((objekt pnavn))
  "Henter objektets navn."
  (navnet objekt))

(defun vis-avsnitt-med-taler (rot)
  "Viser fram et avsnitt med oppgitt taler og katergori."
  (format t "~%~A~%" (vis-avsnitt rot))
  (format t "[node ~D, side ~D]~%" (nodeid rot) (sidetall rot))
  (let ((navnobjekt (nodens-taler rot)))
    (when navnobjekt
      (format t "Taler: ~A. Kategori: ~A.~%-----~%"
              (navnet navnobjekt) (pnavnkategori navnobjekt)))))

(defun vis-alle-avsnitt-med-taler (&optional (forrige (finn-node 1)))
  "Viser fram alle avsnitt med oppgitt taler og kategori."
  (let ((neste (finn-neste-node-av-type forrige '("avsn" "overskr"))))
    (when (and neste
               (< (nodeid neste) *pnavnreg-idstart*))
      (vis-avsnitt-med-taler neste)
      (vis-alle-avsnitt-med-taler (finn-node (+ (nodeid neste) 1))))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Programkjøring                                                      ;
;---------------------------------------------------------------------;

(setf
 *meny*
 '((A (format t "Import/eksport av SGML")
      (1 (format t "Les inn SGML-tekstfila (innhold: ~D noder)."
                 (- (hash-table-count *nodene*)
                    (- *pnavnreg-idslutt*
                       *pnavnreg-idstart*)))
         (progn
           (setq *innfil* (open *innfilnavn-tekst* :direction :input))
           (legg-inn-ny-sgml)
           (setq *filslutt* '())))
      (2 (format t "Les inn SGML-personnavn-registerfila (innhold: ~D noder)."
                 (- *pnavnreg-idslutt*
                    *pnavnreg-idstart*))
```

```
      (progn
        (setq *pnavnreg-idstart* *idnummer*)
        (setq *innfil* (open *innfilnavn-pnavn-reg* :direction :input))
        (legg-inn-ny-sgml)
        (setq *pnavnreg-idslutt* *idnummer*)
        (sett-inn-pnavn-oppslag-noder)
        (setq *pnavnreg-idslutt* *idnummer*)
        (setq *filslutt* '())))
   (3 (format
       t "Sett inn sidetall og nodeid på rett sted (antall sidetall lagt inn: ~D)."
       (hash-table-count *sidetall-nodeid*))
      (progn
        (sett-inn-sidetall-paa-alle-noder)
        (lag-attributtverdier-av-nodeid)))
   (4 (format t "Skriv ut objekttreet som SGML.")
      (skriv-tre-til-fil *utfilnavn*)))
(B (format t "Personregister")
   (1 (format t "Lag nytt navneregister (innhold: ~D navneobjekter)."
              (hash-table-count *navnene*))
      (progn
        (setq *navnnummer* 0)
        (setq *navnene* (make-hash-table))
        (setq *pnavnoppsl* (make-hash-table :test #'equal))
        (lag-navnereg)))
   (2 (format t "Koble navneregister til SGML-tekstfila.")
      (progn
        (legg-inn-navneformer-paa-alle)
        (koble-alle-pnavnreg-til-pnavn)))
   (3 (format t "Vis personnavnobjektene.")
      (skriv-pnavnene))
   (4 (format t "Vis personkategoriene.")
      (skriv-kategoriene))
   (5 (progn
        (format t "Skru ")
        (if *brukerdialog*
            (format t "av")
          (format t "på"))
        (format t " interaksjon."))
      (progn
        (if *brukerdialog*
            (setq *brukerdialog* '())
          (setq *brukerdialog* t))))
   (6 (format t "Nullstill antall interaksjoner (~D til nå)." *foresporsler*)
      (setq *foresporsler* 0))
   (7 (format t "Vis alle taleres avsnittsnumre sortert på talers kategori.")
      (avsn-nummer-kat-fordelt-person))
   (8 (format t "Vis alle avsnittsnumre samlet for hver talerkategori.")
      (avsn-nummer-kat-sum)))
(C (format t "Kobling taler-avsnitt")
   (1 (format t "Sett inn taler på avsnitt som ikke har taler fra før.")
      (progn
        (format
         t "~%Kutt ut avsnittene til og med (blank for å starte med starten): ")
        (time (let ((valg (parse-integer (read-line) :junk-allowed t)))
                (if (and valg (finn-node valg))
                    (velg-taler-til-alle-avsnitt (finn-node valg))
                  (velg-taler-til-alle-avsnitt))))))
```

```
   (2 (format t "Vis alle avsnittene med taler.")
      (vis-alle-avsnitt-med-taler))
   (3 (format t "Endre taler på et avsnitt.")
      (progn
        (format t "~%Endre taler på avsnitt med node nummer: ")
        (let* ((valg (parse-integer (read-line) :junk-allowed t))
               (node (finn-node valg)))
          (if node
              (endre-taler-til-avsnitt node)
            (format t "~%Feil nummer!~%~%")))))
   (4 (format t "Vis statistikk for alle talere.")
      (statistikk-alle-talere)))
(D (format t "Frekvensanalyse")
   (1 (format t "Lag leksikon for alle avsnitt tilordnet taler.")
      (lag-leksikon-med-frekvens-alle-personer))
   (2 (format t "Lag samlet frekvenstabell for alle avsnitt tilordnet taler.")
      (progn
        (lag-standard-leksikon-alle)
        (lag-standard-leksikon-alle-sortert)))
   (3 (format t "Sammenlign en person med snittet. Resultat sortert etter avstand.")
      (sammenlign-person-snitt))
   (4 (format t "Sammenlign en kategori med snittet. Resultat sortert etter avstand.")
      (sammenlign-kategori-snitt))
   (5 (format
       t "Sammenlign en person med snittet. Resultat sortert etter total frekvens.")
      (sammenlign-person-snitt-mest-frekvente))
   (6 (format
       t "Sammenlign en kategori med snittet. Resultat sortert etter total frekvens.")
      (sammenlign-kategori-snitt-mest-frekvente))
   (7 (format
       t "Sammenlign alle kategoriers frekvens med snittet.
Resultat sortert etter total frekvens.")
      (skriv-frekvens-diff-liste (sammenlign-alle-kategorier-snitt-mest-frekvente)))
   (8 (format t "Vis alle kategoriers frekvens. Resultat sortert etter total frekvens.")
      (skriv-frekvens-diff-liste (sammenlign-alle-kategorier-frekvens-mest-frekvente 100)))
   (9 (format t "Vis alle kategoriers sum. Resultat sortert etter total frekvens.")
      (skriv-frekvens-sum-liste (sammenlign-alle-kategorier-sum-mest-frekvente 100)))
   (10 (format t "Vis alle personers frekvens. Resultat sortert etter total frekvens.")
       (skriv-frekvens-diff-liste (sammenlign-alle-navn-frekvens-mest-frekvente)))
   (11 (format t "Vis alle personers ordlengdesnitt.")
       (skriv-snittlengde-ord-alle-frekvenstabeller (snittlengde-ord-alle-frekvenstabeller)))
   (12 (format t "Vis alle personers ordlengdesnitt med signatur.")
       (skriv-snittlengde-signatur-ord-alle-frekvenstabeller
        (lengde-signatur-ord-alle-frekvenstabeller)))
   (13 (format t "Skriv alle personers frekvens i lisp-statistikk-format.
Resultat sortert etter total frekvens. Uaktuelle kategorier utelates.")
       (skriv-frekvens-diff-liste-til-lisp-stat
        (sammenlign-alle-navn-frekvens-mest-frekvente
         101
         '("bonde" "embedsmann" "offiser" "reindriftssame" "sjøsame"))))
   (14 (format t "Skriv alle personers frekvens i lisp-statistikk-format med lagring til fil.
Resultat sortert etter total frekvens. Uaktuelle kategorier utelates.")
       (skriv-frekvens-diff-liste-til-lisp-stat
        (sammenlign-alle-navn-frekvens-mest-frekvente
         101
         '("bonde" "embedsmann" "offiser" "reindriftssame" "sjøsame") )
        :filrot '"a14_500"))
```

```
   (15 (format t "Skriv alle personers frekvens i lisp-statistikk-format.
Resultat sortert etter total frekvens. Uaktuelle kategorier utelates. LoiczView-format.")
       (skriv-frekvens-diff-liste-loicz-view
        (sammenlign-alle-navn-frekvens-mest-frekvente
         101
         '("bonde" "embedsmann" "offiser" "reindriftssame" "sjøsame"))))
   (16 (format t "Skriv alle personers frekvens i SPSS-format.
Resultat sortert etter total frekvens. Uaktuelle kategorier utelates.")
       (skriv-frekvens-diff-liste-spss
        (sammenlign-alle-navn-frekvens-mest-frekvente
         101
         '("bonde" "embedsmann" "offiser" "reindriftssame" "sjøsame")))))
(E (format t "Naboanalyse")
   (1 (format t "Lag kontekstabeller for alle avsnitt tilordnet taler.")
      (lag-kontekst-etter-snavn-tabeller))
   (2 (format t "Lag samlet kontekstabell for alle avsnitt tilordnet taler.")
      (progn
        (lag-etter-snavn-tabeller-alle)
        (lag-etter-snavn-tabeller-alle-sortert)))
   (3 (format t "Vis alle personers etter snavn-frekvens. Resultat sortert etter total
etter snavn-frekvens. Utelater uaktuelle kategorier.")
      (skriv-etter-frekvens-diff-liste
       (sammenlign-alle-navn-etter-tabeller-frekvens-mest-frekvente
        11
        '("bonde" "embedsmann" "offiser" "reindriftssame" "sjøsame"))))
   (4 (format t "Vis alle personers etter snavn-frekvens.
Resultat sortert etter total etter snavn-frekvens.
Utelater uaktuelle kategorier og personer med færre enn 25 kontekster.")
      (skriv-etter-frekvens-diff-liste
       (sammenlign-alle-navn-etter-tabeller-frekvens-mest-frekvente
        11
        '("bonde" "embedsmann" "offiser" "reindriftssame" "sjøsame")
        25)))
   (5 (format t "Skriv alle personers etter snavn-frekvens i lisp-statistikk-format.
Resultat sortert etter total etter snavn-frekvens.
Uaktuelle kategorier utelates.")
      (skriv-frekvens-diff-liste-til-lisp-stat
       (sammenlign-alle-navn-etter-tabeller-frekvens-mest-frekvente
        101
        '("bonde" "embedsmann" "offiser" "reindriftssame" "sjøsame"))))
   (6 (format t "Skriv alle personers etter snavn-frekvens i SPSS-format.
Resultat sortert etter total etter snavn-frekvens.
Uaktuelle kategorier og personer med færre enn 25 kontekster utelates.")
      (skriv-frekvens-diff-liste-spss
       (sammenlign-alle-navn-etter-tabeller-frekvens-mest-frekvente
        11
        '("bonde" "embedsmann" "offiser" "reindriftssame" "sjøsame")
        25)))
   (7 (format t "Skriv etter-kontekst etter snavn + ett eller flere ord for alle personer
sortert etter kategori.")
      (skriv-alle-pnavn-hele-kontekst-etter-snavn
       (progn
         (format t "Skriv null eller flere ord kontekst etter SNAVN, x for å avslutte ")
         (be-om-navneformer))))
   (8 (format t "Skriv mellom-kontekst mellom snavn gruppert på navn,
sortert etter kategori. Maxlengde 10 ord.")
      (skriv-alle-pnavn-hele-kontekst-mellom-snavn 10))
```

```
      (9 (format t "Skriv mellom-kontekst mellom snavn gruppert og sortert
   etter kategori. Maxlengde 10 ord.")
         (skriv-alle-kategorier-hele-kontekst-mellom-snavn 10))
      (10 (format t "Skriv mellom-kontekst mellom snavn gruppert og sortert etter kategori.
   Valgfri maxlengde.")
          (skriv-alle-kategorier-hele-kontekst-mellom-snavn
           (progn
             (format t "~%Maksimalt antall ord: ")
             (parse-integer (read-line) :junk-allowed t)))))
   (F (format t "Vis tekst")
      (1 (format t "Vis teksten under et subtre.")
         (format t "~%~A~%" (vis-avsnitt)))
      (2 (format t "Skriv KWIC-konkordans i HTML-format gruppert etter kategori.")
         (skriv-KWIC-kontekst-alle-kategorier-html))
      (3 (format t "Skriv KWIC-konkordans med pnavn og snavn erstattet i HTML-format
   gruppert etter kategori.")
         (skriv-KWIC-kontekst-alle-kategorier-html :ikke-noder '("pnavn" "snavn")))
      (4 (format t "Skriv KWIC-konkordans med trunkering av ordet
   med pnavn og snavn erstattet i HTML-format gruppert etter kategori.")
         (skriv-KWIC-kontekst-alle-kategorier-html :trunk t :ikke-noder '("pnavn" "snavn")))
      (5 (format t "Skriv KWIC-konkordans i HTML-format sortert etter nodenummer.")
         (skriv-KWIC-kontekst-all-tekst-html)))
   (G (format t "Grunner")
      (1 (format t "Skriv ut alle grunner.")
         (skriv-alle-grunner))
      (2 (format t "Skriv ut grunnene med tilknytning til en bestemt node.")
         (skriv-node-grunn '()))
      (3 (format t "Skriv ut alle grunner som er knyttet til avsnitt uttalt av talere
   av en kategori.")
         (skriv-grunner-noder-med-taler-kat '())))
   (H (format t "Lagring")
      (1 (format t "Lagre datastrukturen.")
         (time (lagre)))
      (2 (format t "Les inn lagret datastruktur.")
         (time (les))))))


(defun kjoer ()
  "Kommandoskall med menyer. Leser fila definert over."
  (format t "~%~%HOVEDMENY~%")
  (mapc #'(lambda (x)
            (format t "~%~A. " (car x))
            (eval (cadr x)))
        *meny*)
  (format t "~%Q. Avslutt~%~%Valg: ")
  (let* ((retur (read))
         (valg (assoc retur *meny*)))
    (cond (valg
           (undermeny (cdr valg)))
          ((eq retur 'q)
           '())
          (t
           (kjoer)))))

(defun undermeny (innhold)
  "Undermeny med valg"
  (format t "~%~%DELMENY: ")
  (eval (car innhold))
```

```
  (format t "~%")
  (mapc #'(lambda (x)
            (format t "~%~A. " (car x))
            (eval (cadr x)))
        (cdr innhold))
  (format t "~%Q. Ut~%~%Valg: ")
  (let* ((retur (read))
         (valg (assoc retur (cdr innhold))))
    (cond (valg
           (eval (caddr valg))
           (undermeny innhold))
          ((eq retur 'q)
           (kjoer))
          (t
           (undermeny innhold)))))


(load "grunner-1.0.0.lsp")
(load "hjelpefunk-1.1.0.lsp")
(load "analyser-1.0.0.lsp")
```

A.3 analyser

```
;;=============================================================
;; DEFINISJONER
;;-------------------------------------------------------------

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Globale variable                                            ;
;-------------------------------------------------------------;

(setq *frekvenstabeller* (make-hash-table)) ; Holder frekvenstabellene
                                            ; for alle personer med
                                            ; avsnitt.
;;; Hvert oppslag har et par på formen (totalt-antall-ord .
;;; frekvenstabell) der frekvenstabellen er en hashtabell med alle
;;; ordfrekvensene

(setq *frekvenstabell-alle* (make-hash-table)) ; Total frekvenstabell for
                                               ; alle anlyserbare talere.

(setq *frekvenstabell-alle-sort* '())   ; Total frekvenstabell for
                                        ; alle anlyserbare talere
                                        ; sortert etter frekvens.


(setq *kontekst-etter-snavn-tabeller* (make-hash-table))
                                        ; Holder frekvenstabellene for
                                        ; forekomster etter snavn for
                                        ; alle personer med avsnitt.
;;; Hvert oppslag har et par på formen (totalt-antall-ord .
;;; frekvenstabell) der frekvenstabellen er en hashtabell med
;;; ordfrekvensene

(setq *kontekst-etter-snavn-tabeller-alle* (make-hash-table))
                                        ; Total etter snavn-
                                        ; frekvenstabell for alle
                                        ; anlyserbare talere.

(setq *kontekst-etter-snavn-tabeller-alle-sort* '())
                                        ; Total etter navn-
                                        ; frekvenstabell for alle
                                        ; anlyserbare talere sortert
                                        ; etter frekvens.


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Hjelperutiner                                               ;
;-------------------------------------------------------------;

(defun sett-inn-kat-nr (liste)
  "Returnerer nummer basert på kategorinavn i ei liste."
  (let ((nrlis '(("adel" 1)
                 ("arbeider" 2)
                 ("bonde" 3)
                 ("embedsmann" 4)
                 ("fastboende" 5)
```

```
                ("funksjonær" 6)
                ("håndverker" 7)
                ("jeger" 8)
                ("kirke" 9)
                ("kvæn" 10)
                ("lagrett" 11)
                ("offiser" 12)
                ("politi" 13)
                ("reindriftssame" 14)
                ("sjøsame" 15)
                ("skole" 16)
                ("soldat" 17))))
    (mapcar #'(lambda (x)
                (let ((treff (assoc x nrlis :test #'equal-case)))
                  (if treff
                      (cadr treff)
                    18)))
            liste)))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Tell noder.                                                          ;
;----------------------------------------------------------------------;

(defun tell-snavn-i-subtre (rotnodeliste)
  "Returnerer antall snavn-noder i subtreet."
  (length (finn-noder-i-subtre rotnodeliste '"snavn")))


(defun etter-snavn-i-alle-kat ()
  "Returnerer et sett av lister som inneholder frekvensene for alle
   kategoriene."
  (let ((hovedliste '()))
    (maphash #'(lambda (id kat)
                 (unless (null (kat-noder kat))
                   (push (list (kategori kat)
                               (reduce #'+ (mapcar #'length
                                                   (mapcar #'vis-avsnitt
                                                           (kat-noder kat))))
                               (etter-snavn-i-subtre (kat-noder kat)))
                         hovedliste)
                   (format t "~%Kategori ~A..." (kategori kat))))
             *kategori-statistikk*)
    hovedliste))

(defun etter-snavn-i-subtre (rotnodeliste)
  "Returnerer første pcdata-streng etter hvert snavn i subtrærne
   definert i rotnodeliste."
  (let ((nodene (finn-noder-i-subtre rotnodeliste '"snavn"))
        (resultat (make-hash-table :test #'equal)))
    (mapcan #'(lambda (x)
                (let ((resten (sosken-etter x)))
                  (when resten
                    (let* ((hoyre (hoyre-kontekst resten))
                           (oppslag (gethash hoyre resultat)))
                      (setf (gethash hoyre resultat)
                            (if oppslag
```

```
                              (incf oppslag)
                            1))))))
          nodene)
    (sort (hash2list resultat) #'> :key #'cdr)))

(defun hoyre-kontekst (nodeliste)
  "Plukker første element i nodeliste som er aktuelt."
  (if nodeliste
      (let ((forste (car nodeliste)))
        (if (typep forste 'pcdata)
            (let ((ordene (hent-ord (pcdatainnhold forste))))
              (cond ((and ordene (equal-case (car ordene) '"og"))
                     (concatenate 'string (car ordene) '" "
                                  (hoyre-kontekst (cdr nodeliste))))
                    ((and ordene (not (equal-case (car ordene) '"mdash")))
                     (car ordene))
                    (t
                     (hoyre-kontekst (cdr nodeliste)))))
          (let ((snavn (finn-noder-i-subtre (list forste) '"snavn"))
                (pnavn (finn-noder-i-subtre (list forste) '"pnavn")))
            (cond (snavn
                   '"<snavn>")
                  (pnavn
                   '"<pnavn>")
                  (t
                   (hoyre-kontekst (cdr nodeliste)))))))
    '"<tomt>"))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Statistikk over avsnitt                                              ;
;----------------------------------------------------------------------;

(defun lag-statistikk-alle-talere ()
  "Bygger opp statistikkobjektene med informasjon fra node- og
   pnavn-objektene"
  (let ((kategoriene (let ((liste '()))
                       (push '"" liste)
                       (maphash #'(lambda (x y)
                                    (push y liste))
                                *tittel-kategori*)
                       (sort (remove-duplicates liste :test #'equal)
                             #'string<))))
    (mapcan #'(lambda (kategorien)
                (let ((kat-navnene '())
                      (kat-nodene '()))
                  (maphash #'(lambda (id navn)
                               (when (equal kategorien
                                            (pnavnkategori navn))
                                 (push navn kat-navnene)
                                 (mapcar #'(lambda (node)
                                             (push node kat-nodene))
                                         (pnavntaleravsn navn))))
                           *navnene*)
                  (setf (gethash kategorien *kategori-statistikk*)
                        (make-instance
                         'kategori
```

```
                        :id (incf *katid*)
                        :kat kategorien
                        :kat-navn kat-navnene
                        :kat-noder kat-nodene))))
           kategoriene)))

(defun skriv-kategori-statistikk (&optional (pnavnp '()))
  "Statistikk 1: Skriver ut statistikk basert på informasjon i kategori-objektene."
  (maphash #'(lambda (id kat)
               (unless (null (kat-noder kat))
                 (let ((kat-navn (kat-navn kat)))
                   (format t "~%~%Kategori ~D: ~A" (katid kat) (kategori kat))
                   (when pnavnp
                     (mapcar #'skriv-statistikk-taler kat-navn))
                   (skriv-statistikk-kat kat)
                   (format t "~%====================~%")
                   )))
           *kategori-statistikk*))

(defun finn-pnavn-som-taler ()
  "Returnerer alle pnavn-objekter som er taler til minst ett avsnitt."
  (let ((utliste '()))
    (maphash #'(lambda (x y)
                 (when (pnavntaleravsn y)
                   (push y utliste)))
             *navnene*)
    (reverse utliste)))

(defun skriv-statistikk-taler (pnavn)
  "Skriver taler-statistikk for navn-objektet. Returnerer personens sammendrag."
  (declare (fixnum ant-avsn sum-lengde snittlengde))
  (when (and (typep pnavn 'pnavn)
             (pnavntaleravsn pnavn))
    (format t "~%Navn ~D: ~A"
            (navnid pnavn)
            (navnet pnavn))
    (let* ((kategori (pnavnkategori pnavn))
           (avsnittene (pnavntaleravsn pnavn))
           (ant-avsn (length avsnittene))
           (avsn-tekstene (mapcar #'vis-avsnitt avsnittene))
           (avsn-lengder (mapcar #'length avsn-tekstene))
           (sum-lengde (reduce #'+ avsn-lengder))
           (snittlengde (round (/ sum-lengde ant-avsn)))
           (ant-snavn (length (finn-noder-i-subtre avsnittene '"snavn")))
           (tegn-pr-snavn (round (if (< 0 ant-avsn)
                                     (/ sum-lengde ant-avsn)
                                   99999))))
      (unless (tom-strengp kategori)
        (format t " (~A)" kategori))
      (format t "~%Antall avsnitt: ~D. Lengde: ~D tegn. Snittlengde: ~D tegn.
   Tegn pr. snavn: ~D.~%-----~%"
              ant-avsn sum-lengde snittlengde tegn-pr-snavn)
      (cons kategori avsn-tekstene))))

(defun skriv-statistikk-kat (kat)
  "Skriver taler-statistikk for kategori-objektet."
  (declare (fixnum ant-avsn sum-lengde snittlengde))
```

```
  (when (and (typep kat 'kategori)
             (kat-noder kat))
    (let* ((kategori (kategori kat))
           (avsnittene (kat-noder kat))
           (ant-avsn (length avsnittene))
           (avsn-tekstene (mapcar #'vis-avsnitt avsnittene))
           (avsn-lengder (mapcar #'length avsn-tekstene))
           (sum-lengde (reduce #'+ avsn-lengder))
           (snittlengde (round (/ sum-lengde ant-avsn)))
           (ant-snavn (length (finn-noder-i-subtre avsnittene '"snavn")))
           (tegn-pr-snavn (round (/ sum-lengde ant-snavn))))
      (format t "~%Sum for kategori ~A:" kategori)
      (format t "~%Antall avsnitt: ~D. Snittlengde: ~D tegn. Tegn pr. snavn: ~D.~%-----~%"
              ant-avsn snittlengde tegn-pr-snavn))))

(defun sorter-kategori (liste)
  "Tar imot ei liste av (kategori tekst tekst ...) og sorterer på kategori."
  (let ((kategoriliste (make-hash-table :test #'equal)))
    (dolist (katlis liste)
            (setf (gethash (car katlis) kategoriliste)
                  (append (gethash (car katlis) kategoriliste) (cdr katlis))))
    kategoriliste))

(defun statistikk-alle-talere ()
  "Skriver taler-statistikk for alle talere. SKRIV OM"
  (let ((utliste '()))
    (dolist (pnavn (finn-pnavn-som-taler))
            (push (skriv-statistikk-taler pnavn) utliste))
    (let ((kategori-tekst (sorter-kategori utliste)))
      (maphash
       #'(lambda (kategori strengene)
           (let* ((ant-avsn (length strengene))
                  (avsn-lengder (mapcar #'length strengene))
                  (sum-lengde (reduce #'+ avsn-lengder))
                  (snittlengde (round (if (> 0 ant-avsn)
                                          (/ sum-lengde ant-avsn)
                                        99999))))
             (format t "~%Kategori ~A: Antall avsnitt: ~D. Lengde: ~D tegn.
   Snittlengde: ~D tegn."
                     kategori ant-avsn sum-lengde snittlengde)))
       kategori-tekst)
      (format t "~%~%"))))

(defun lag-etter-snavn-sammendrag (liste)
  "Statistikk 2: Returnerer sammendrag av etter-snavn for alle kategorier."
  (let ((full-tabell (make-hash-table :test #'equal))
        (kat-liste (mapcar #'car liste)))
    (dolist (kat-full liste)
            (let ((kat (car kat-full))
                  (lengde (cadr kat-full))
                  (frekvenser (caddr kat-full)))
              (format t "~%Kategori ~A gås gjennom..." kat)
              (mapc #'(lambda (ordpar)
                        (let* ((ord (car ordpar))
                               (ant (cdr ordpar))
                               (oppslag (gethash ord full-tabell)))
                          (setf (gethash ord full-tabell)
```

```
                              (cons (cons kat (float (/ ant lengde))) oppslag))))
                    frekvenser)))
    (format t "~%~%'Frekvenser:'~%~%~{  '~A'~}  ~A" kat-liste '"'snitt'")
    (dolist (ordliste (hash2list full-tabell))
            (format t "~%'~A'" (car ordliste))
            (dolist (kategori kat-liste)
                    (let* ((frekvp (cdr (assoc ‘,kategori (cdr ordliste))))
                           (frekv  (if frekvp
                                       frekvp
                                     0)))
                      (format t "       ~F" (* 10000 frekv))))
            (format t " ~F" (* 10000 (/ (reduce #'+ (mapcar #'cdr (cdr ordliste)))
                                        (length kat-liste)))))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Frekvensanalyse                                                      ;
;----------------------------------------------------------------------;

(defun skriv-ut-personer-med-leksikon-oppsummering ()
  "Skriver ut alle personene som har leksikon med litt statistikk."
  (maphash #'(lambda (id frekvenspar)
               (let ((pnavn (finn-navn id)))
                 (skriv-pnavn-kort pnavn)
                 (format t "~%Antall ord i leksikon: ~D.~%~%"
                         (hash-table-count (cdr frekvenspar)))))
           *frekvenstabeller*))


(defun lag-leksikon-med-frekvens-alle-personer ()
  "Lager leksikon for avsnittene til alle personer."
  (clrhash *frekvenstabeller*)
  (maphash
   #'(lambda (id pnavn)
       (let ((avsnitt (pnavntaleravsn pnavn)))
         (when avsnitt
           (setf (gethash id *frekvenstabeller*)
                 (skriv-leksikon-med-frekvens avsnitt))
           (format t "Skrevet pnavn ~D, ~D avsnitt.~%" id (length avsnitt)))))
   *navnene*))

(defun skriv-leksikon-med-frekvens (nodeliste)
  "Skriver leksikon for nodene med røtter i nodeliste basert på pcdata. Skalerer."
  (let ((frekvensliste (make-hash-table :test #'equal))
        (nodenrpar (mapcar #'(lambda (node)
                               (let* ((id-fra (nodeid node))
                                      (id-til (finn-siste-nodeid node)))
                                 (cons id-fra id-til)))
                           nodeliste))
        (pcdata '""))
    (dolist (par nodenrpar)
      (setq pcdata
            (concatenate 'string pcdata
                         (finn-pcdata-fra-til (car par)
                                              (cdr par)
                                              :ikke-noder '("pnavn" "snavn")))))
    (let ((ordliste (fjern-streng-fra-liste
                     '"mdash"
```

```
                    (hent-ord-s (string-downcase (rydd-streng pcdata))
                                '"1234567890-/"))))
      (dolist (ord ordliste)
        (let ((frekvens (gethash ord frekvensliste)))
          (setf (gethash ord frekvensliste)
                (if frekvens
                    (+ frekvens 1)
                    1))))
      (cons (length ordliste) frekvensliste))))

(defun skriv-standard-leksikon (&optional (kategori '()))
  "Lager et leksikon for alle avsnitt tilordnet aktuelle talere av
   oppgitt kategori (ingen parameter gir alle). Det betyr at 'skal ikke
   analyseres' kuttes ut."
  (let ((leksikonet (make-hash-table :test #'equal))
        (antall 0))
    (maphash #'(lambda (id frekvenspar)
                 (when (not (= id 3)) ; Kutter ut "Skal ikke analyseres"
                   (let* ((pnavn (finn-navn id))
                          (pnavnkat (pnavnkategori pnavn)))
                     (when (or (not kategori)
                               (equal-case pnavnkat kategori))
                       (dolist (frekvpar (hash2list (cdr frekvenspar)))
                               (let* ((ord (car frekvpar))
                                      (frekvens (cdr frekvpar))
                                      (frekv-til-naa (gethash ord leksikonet)))
                                 (setf (gethash ord leksikonet)
                                       (if frekv-til-naa
                                           (+ frekvens frekv-til-naa)
                                         frekvens))))
                       (setq antall (+ antall (car frekvenspar)))
                       (format t "Lagt til navn ~D...~%" id)))))
             *frekvenstabeller*)
    (cons antall leksikonet)))

(defun lag-standard-leksikon-alle ()
  "Lager et leksikon med frekvenser for alle talere."
  (setq *frekvenstabell-alle*
        (skriv-standard-leksikon))
  'gjort)

(defun lag-standard-leksikon-alle-sortert ()
  "Lager et leksikon med frekvenser for alle talere sortert etter frekvens."
  (setq *frekvenstabell-alle-sort*
        (sort (hash2list (cdr *frekvenstabell-alle*))
              #'(lambda (x y) (> (cdr x) (cdr y)))))
  'gjort)

(defun skriv-ut-leksikon (lexpar &optional (antall '()))
  "Skriver ut leksikonet med mest frekvente ord først. Lexpar er på
   formen (totalt-antall-ord . frekvenstabell)"
  (let* ((lex (cdr lexpar))
         (antord (car lexpar))
         (innliste (sort (hash2list lex)
                         #'(lambda (x y) (> (cdr x) (cdr y))))))
    (dolist (frekvpar (if antall
                          (subseq innliste 0 antall)
```

```
                        innliste))
      (format t "~%~,5,,,F: ~A" (/ (cdr frekvpar) antord) (car frekvpar)))))

(defun skriv-leksikon-differanse (lexpar1 lexpar2 &optional (antall '()))
  "Skriver snittet av leksikonene sortert etter synkende absolutt
   avstand. lex1-lex2. Mest effektivt å ta minste leksikon som første
   parameter"
  (let ((lex1 (cdr lexpar1))
        (antord1 (car lexpar1))
        (lex2 (cdr lexpar2))
        (antord2 (car lexpar2))
        (utliste-pluss '())
        (utliste-minus '())
        (max (if antall
                 (floor (/ antall 2))
               '())))
    (maphash #'(lambda (ord1 frekvens1)
                 (let ((frekvens2 (gethash ord1 lex2)))
                   (when frekvens2
                     (let ((diff (- (/ frekvens1 antord1)
                                    (/ frekvens2 antord2))))
                       (push (cons ord1 diff)
                             (if (> diff 0)
                                 utliste-pluss
                               utliste-minus))))))
             lex1)
    (format t "~%Overrepresentasjon i argument 1:")
    (let ((utliste-sort (sort utliste-pluss #'(lambda (x y)
                                                (> (abs (cdr x))
                                                   (abs (cdr y)))))))
      (dolist (diffpar (if (and antall (> (length utliste-sort) max))
                           (subseq utliste-sort 0 max)
                         utliste-sort))
        (format t "~%Avstand ~,5,,,F: ~A" (cdr diffpar) (car diffpar))))
    (format t "~%~%Underrepresentasjon i argument 1:")
    (let ((utliste-sort (sort utliste-minus #'(lambda (x y)
                                                (> (abs (cdr x))
                                                   (abs (cdr y)))))))
      (dolist (diffpar (if (and antall (> (length utliste-sort) max))
                           (subseq utliste-sort 0 max)
                         utliste-sort))
        (format t "~%Avstand ~,5,,,F: ~A" (cdr diffpar) (car diffpar))))
    (format
     t "~%~%Antall ord i argument 1: ~D; argument 2: ~D~%--------------------------"
     antord1 antord2)))

(defun spor-etter-pnavn ()
  "Spør brukeren etter et pnavnnummer, evt. søk."
  (format t "~%Skriv inn pnavnnummer eller blank for søk på navnet: ")
  (let ((input (read-line)))
    (if (or (not input) (equal input ""))
        (sok-etter-taler '" *")
      (let ((pnavn (finn-navn (parse-integer input :junk-allowed t))))
        (if pnavn
            pnavn
          (progn
            (format t "~%Fant ikke navnet.~%")
```

```
           (spor-etter-pnavn)))))))

(defun sammenlign-person-snitt (&optional (navn '()))
  "Sammenligner en person med snittet og returerer sortert etter
   største avstand. Hvis ikke node som argument, spørres det."
  (when (null navn)
    (setq navn (spor-etter-pnavn)))
  (let ((frekvenspar (gethash (navnid navn) *frekvenstabeller*)))
    (if frekvenspar
        (progn
          (format t "~%~%Sammenligning av person ~D (~A) med snittet:~%"
                  (navnid navn) (navnet navn))
          (skriv-leksikon-differanse frekvenspar *frekvenstabell-alle* 20))
      (format t "~%Navnet ~D har ikke tilknyttet avsnitt.~%" (navnid navn)))))

(defun spor-etter-kat ()
  "Spør brukeren etter en kategori."
  (format t "~%Disse kategoriene finnes:~%")
  (let ((alternativer (let ((liste '()))
                        (maphash #'(lambda (x y)
                                     (push y liste))
                                 *tittel-kategori*)
                        (sort (remove-duplicates liste :test #'equal) #'string<))))
    (skriv-alternativer alternativer '"Hvilken kategori? ")
    (let ((input (parse-integer (read-line) :junk-allowed t)))
      (if (and input (< 0 input) (>= (length alternativer) input))
          (elt alternativer (- input 1))
        (spor-etter-kat)))))

(defun sammenlign-kategori-snitt (&optional (kat '()))
  "Sammenligner en kategori med snittet og returerer sortert etter
   største avstand. Hvis ikke node som argument, spørres det."
  (when (null kat)
    (setq kat (spor-etter-kat)))
  (let ((frekvenspar (skriv-standard-leksikon kat)))
    (if (= (car frekvenspar) 0)
        (format t "~%Navn med kategori ~A har ikke tilknyttet avsnitt.~%" kat)
      (progn
        (format t "~%~%Sammenligning av personer i kategori ~A med snittet:~%" kat)
        (skriv-leksikon-differanse frekvenspar *frekvenstabell-alle* 20)))))

(defun skriv-leksikon-differanse-mest-frekvente (lexpar1 lexpar2 &optional (antall '()))
  "Skriver snittet av leksikonene sortert etter synkende total
  frekvens."
  (let ((ord-analyse (if antall
                         (subseq *frekvenstabell-alle-sort* 0 (- antall 1))
                       *frekvenstabell-alle-sort*))
        (lex1 (cdr lexpar1))
        (antord1 (car lexpar1))
        (lex2 (cdr lexpar2))
        (antord2 (car lexpar2))
        (utliste '()))
    (mapc #'(lambda (frekvenspar)
              (let* ((ord (car frekvenspar))
                     (frekvens (cdr frekvenspar))
                     (f1 (gethash ord lex1))
                     (f2 (gethash ord lex2))
```

```
                    (frekvens1 (if f1 f1 0))
                    (frekvens2 (if f2 f2 0))
                    (diff (- (/ frekvens1 antord1)
                             (/ frekvens2 antord2))))
               (push (cons ord diff) utliste)))
          ord-analyse)
    (format t "~%Overrepresentasjon i argument 1:")
    (dolist (diffpar (reverse utliste))
            (format t "~%Avstand ~,5,,,F: ~A" (cdr diffpar) (car diffpar)))
    (format
     t "~%~%Antall ord i argument 1: ~D; argument 2: ~D~%-------------------------"
     antord1 antord2)))

(defun skriv-frekvens-diff-liste (diffliste)
  "Skriver ut ei diffliste som er på formen (navn antall-ord (differanser))."
  (format t "~%                          ~{~10,,,@A ~}"
          (mapcar #'(lambda (x)
                      (let ((navn (car x)))
                        (if (> (length navn) 8)
                            (subseq navn 0 9)
                          navn)))
                  diffliste))
  (format t "~%           ~10,,,D " (car *frekvenstabell-alle*))
  (format t "~{~10,,,D ~}" (mapcar #'cadr diffliste))
  (let ((lengde (length diffliste)))
    (dolist (ordpar (caddar diffliste))
            (format t "~%~10,,,A ~10,,,D "
                    (car ordpar)
                    (gethash (car ordpar) (cdr *frekvenstabell-alle*)))
            (dotimes (teller lengde)
                     (let* ((ordliste (caddr (elt diffliste teller)))
                            (indre-ordpar (assoc (car ordpar) ordliste
                                                 :test #'equal)))
                       (format t "~,7,,,@F " (cdr indre-ordpar)))))))

(defun skriv-frekvens-diff-liste-til-lisp-stat (diffliste &key filrot)
  "Skriver ut ei diffliste som er på formen (navnid antall-ord
   (differanser)) til et format som kan lese av lisp-stat. Hvis filrot
   er oppgitt, skrives også til fil."
  (format t "~%(defun skriv-plots ()")
  (format t "~%(def kat (list~{ ~D~}))"
          (sett-inn-kat-nr (mapcar #'(lambda (x)
                                       (pnavnkategori
                                        (finn-navn
                                         (parse-integer (car x)))))
                                   diffliste)))
  (let ((lengde (length diffliste)))
    (dolist (ordpar (caddar diffliste))
      (let* ((ordet (car ordpar))
             (ord-navn (cond ((not ordet)
                              '"SISTE_ORD")
                             ((parse-integer ordet :junk-allowed t)
                              (concatenate 'string '"TALL_" ordet))
                             (t
                              (byttut '"ae" '"æ"
                                      (byttut '"oe" '"ø" ordet))))))
        (format t "~%(def ~A (list" ord-navn)
```

```
        (dotimes (teller lengde)
          (let* ((ordliste (caddr (elt diffliste teller)))
                 (indre-ordpar (assoc (car ordpar) ordliste :test #'equal)))
            (format t " ~,7,,,F" (cdr indre-ordpar))))
        (format t "))")
        (format
         t "~%(def plottene (plot-points kat ~A :variable-labels '(~Ckategori~C ~Cfrekvens~C)))"
         ord-navn #\" #\" #\" #\")
        (format t "~%(send plottene :title ~C~A~C)" #\" ordet #\")
        (format t "~%(send plottene :add-points kat ~A)" ord-navn)
        (format t "~%(send plottene :draw-text ~C~C~A~C 200 10 1 1)" #\' #\" ordet #\")
        (format t "~%(send plottene :variable-label 1 ~C~Cfrekvens ~A~C)" #\' #\" ordet #\")
        (if filrot
            (progn
              (format t "~%(send plottene :save-image ~C~C~Aanalyse-data/~A~A.ps~C)"
                      #\' #\" '"/hf/hedvig/muspro-u1/oeide/utv/lisp/hoppg/"
                      filrot (byttut '"_skraa_" '"/" ord-navn) #\")
              (format t "~%(send plottene :close)")))))
    (format t ")")))

(defun skriv-frekvens-diff-liste-spss (diffliste)
  "Skriver ut ei diffliste som er på formen (navn antall-ord
   (differanser)) i format for spss, dvs. at hver person har ei linje og
   hvert ord ei kolonne."
  (format t "~%§ID§|§navn§|§katnr§|§kat§~{|§~A§~}" (mapcar #'car (caddar diffliste)))
  (dolist (person diffliste)
          (let ((pnavn (finn-navn (parse-integer (car person)))))
            (format t "~%~D|§~A§|~D|§~A§~{|~,7,,,F~}"
                    (parse-integer (car person))
                    (byttut '"_" '"," (navnet pnavn))
                    (car (sett-inn-kat-nr (list (pnavnkategori pnavn))))
                    (pnavnkategori pnavn)
                    (mapcar #'cdr (caddr person))))))

(defun skriv-frekvens-diff-liste-loicz-view (diffliste)
  "Skriver ut ei diffliste som er på formen (navn antall-ord
   (differanser)) i format for loicz-view."
  (format t "~%§ID§,§LONG§,§LAT§,§!navn§~{,§~A§~}" (mapcar #'car (caddar diffliste)))
  (dolist (person diffliste)
          (format t "~%~D,~D,~D,§~A§~{,~,7,,,F~}"
                  (parse-integer (car person))
                  0
                  0
                  (byttut '"_" '"," (navnet (finn-navn (parse-integer (car person)))))
                  (mapcar #'cdr (caddr person)))))

(defun skriv-frekvens-sum-liste (sumliste)
  "Skriver ut ei sumliste som er på formen (navn antall-ord (summer))."
  (format t "~%                      ~{~10,,,@A ~}"
          (mapcar #'(lambda (x)
                      (let ((navn (car x)))
                        (if (> (length navn) 8)
                            (subseq navn 0 9)
                          navn)))
                  sumliste))
  (format t "~%           ~10,,,D " (car *frekvenstabell-alle*))
  (format t "~{~10,,,D ~}" (mapcar #'cadr sumliste))
```

```
  (let ((lengde (length sumliste)))
    (dolist (ordpar (caddar sumliste))
      (format t "~%~10,,,A ~10,,,D "
              (car ordpar)
                    (gethash (car ordpar) (cdr *frekvenstabell-alle*)))
      (dotimes (teller lengde)
        (let* ((ordliste (caddr (elt sumliste teller)))
               (indre-ordpar (assoc (car ordpar) ordliste :test #'equal)))
          (format t "~10,,,D " (cdr indre-ordpar)))))))

(defun returner-leksikon-differanse-mest-frekvente (lexpar1
                                                    lexpar2
                                                    &optional (antall '()))
  "Returnerer snittet av leksikonene sortert etter synkende total
   frekvens."
  (let ((ord-analyse (if antall
                         (subseq *frekvenstabell-alle-sort* 0 (- antall 1))
                       *frekvenstabell-alle-sort*))
        (lex1 (cdr lexpar1))
        (antord1 (car lexpar1))
        (lex2 (cdr lexpar2))
        (antord2 (car lexpar2))
        (utliste '()))
    (mapc #'(lambda (frekvenspar)
              (let* ((ord (car frekvenspar))
                     (frekvens (cdr frekvenspar))
                     (f1 (gethash ord lex1))
                     (f2 (gethash ord lex2))
                     (frekvens1 (if f1 f1 0))
                     (frekvens2 (if f2 f2 0))
                     (diff (- (/ frekvens1 antord1)
                              (/ frekvens2 antord2))))
                (push (cons ord diff) utliste)))
          ord-analyse)
    (reverse utliste)))

(defun returner-leksikon-sum-mest-frekvente (lexpar &optional (antall '()))
  "Returnerer summen av leksikonet sortert etter synkende total
   frekvens."
  (let ((ord-analyse (if antall
                         (subseq *frekvenstabell-alle-sort* 0 (- antall 1))
                       *frekvenstabell-alle-sort*))
        (lex (cdr lexpar))
        (antord (car lexpar))
        (utliste '()))
    (mapc #'(lambda (frekvenspar)
              (let* ((ord (car frekvenspar))
                     (f (gethash ord lex))
                     (frekvens (if f f 0)))
                (push (cons ord frekvens) utliste)))
          ord-analyse)
    (reverse utliste)))

(defun returner-leksikon-frekvens-mest-frekvente (lexpar &optional (antall '()))
  "Returnerer frekvensen av leksikonet sortert etter synkende total
   frekvens."
  (let ((ord-analyse (if antall
```

```
                      (subseq *frekvenstabell-alle-sort* 0 (- antall 1))
                    *frekvenstabell-alle-sort*))
       (lex (cdr lexpar))
       (antord (car lexpar))
       (utliste '()))
   (mapc #'(lambda (frekvenspar)
             (let* ((ord (car frekvenspar))
                    (f (gethash ord lex))
                    (frekvens (if f f 0)))
               (push (cons ord (/ frekvens antord)) utliste)))
         ord-analyse)
   (reverse utliste)))

(defun returner-leksikon-frekvens-ett-ord (lexpar ord)
  "Returnerer frekvensen av ett ord i leksikonet."
  (let* ((antord (car lexpar))
         (lex (cdr lexpar))
         (frekv (gethash ord lex))
         (frekv-ut (if frekv
                       frekv
                     0)))
    (list (cons ord (/ frekv-ut antord)))))

(defun sammenlign-person-snitt-mest-frekvente (&optional (navn '()))
  "Sammenligner en person med snittet og returerer sortert etter
   total frekvens. Hvis ikke navn som argument, spørres det."
  (when (null navn)
    (setq navn (spor-etter-pnavn)))
  (let ((frekvenspar (gethash (navnid navn) *frekvenstabeller*)))
    (if frekvenspar
        (progn
          (format t "~%~%Sammenligning av person ~D (~A) med snittet:~%"
                  (navnid navn) (navnet navn))
          (skriv-leksikon-differanse-mest-frekvente frekvenspar
                                                    *frekvenstabell-alle*
                                                    20))
      (format t "~%Navnet ~D har ikke tilknyttet avsnitt.~%" (navnid navn)))))

(defun sammenlign-kategori-snitt-mest-frekvente (&optional (kat '()))
  "Sammenligner en kategori med snittet og returerer sortert etter
   total frekvens. Hvis ikke kat som argument, spørres det."
  (when (null kat)
    (setq kat (spor-etter-kat)))
  (let ((frekvenspar (skriv-standard-leksikon kat)))
    (if (= (car frekvenspar) 0)
        (format t "~%Navn med kategori ~A har ikke tilknyttet avsnitt.~%" kat)
      (progn
        (format t "~%~%Sammenligning av personer i kategori ~A med snittet:~%" kat)
        (skriv-leksikon-differanse-mest-frekvente frekvenspar
                                                  *frekvenstabell-alle*
                                                  20)))))

(defun sammenlign-alle-kategorier-snitt-mest-frekvente (&optional (antall 50)
                                                                  (kategoriliste '()))
  "Sammenligner alle kategori med snittet og returerer sortert etter
   total frekvens. Kan velge å analysere bare et utvalg kategorier."
  (let* ((kat (let ((liste '()))
```

```
               (maphash #'(lambda (x y)
                            (when (or (null kategoriliste)
                                      (search y kategoriliste :test #'equal-case))
                              (push y liste)))
                        *tittel-kategori*)
               (sort (remove-duplicates liste :test #'equal) #'string<)))
        (kat-liste (mapcar #'(lambda (x)
                               (cons x (skriv-standard-leksikon x)))
                           kat))
        (kat-liste-avsn (filter #'(lambda (x)
                                    (when (not (= (cadr x) 0))
                                      x))
                                kat-liste))
        (utliste '()))
    (dolist (triplet kat-liste-avsn)
            (push (list (car triplet)
                        (cadr triplet)
                        (returner-leksikon-differanse-mest-frekvente
                         (cdr triplet)
                         *frekvenstabell-alle*
                         antall))
                  utliste))
    utliste))

(defun sammenlign-alle-kategorier-sum-mest-frekvente (&optional (antall 50))
  "Regner ut summen for alle kategorier og returerer sortert etter
   total frekvens."
  (let* ((kat (let ((liste '()))
               (maphash #'(lambda (x y)
                            (push y liste))
                        *tittel-kategori*)
               (sort (remove-duplicates liste :test #'equal) #'string<)))
        (kat-liste (mapcar #'(lambda (x)
                               (cons x (skriv-standard-leksikon x)))
                           kat))
        (kat-liste-avsn (filter #'(lambda (x)
                                    (when (not (= (cadr x) 0))
                                      x))
                                kat-liste))
        (utliste '()))
    (dolist (triplet kat-liste-avsn)
            (push (list (car triplet)
                        (cadr triplet)
                        (returner-leksikon-sum-mest-frekvente (cdr triplet)
                                                              antall))
                  utliste))
    utliste))

(defun sammenlign-alle-kategorier-frekvens-mest-frekvente (&optional (antall 50))
  "Regner ut frekvensen for alle kategorier og returerer sortert etter
   total frekvens."
  (let* ((kat (let ((liste '()))
               (maphash #'(lambda (x y)
                            (push y liste))
                        *tittel-kategori*)
               (sort (remove-duplicates liste :test #'equal) #'string<)))
        (kat-liste (mapcar #'(lambda (x)
```

```
                              (cons x (skriv-standard-leksikon x)))
                            kat))
        (kat-liste-avsn (filter #'(lambda (x)
                                    (when (not (= (cadr x) 0))
                                      x))
                                kat-liste))
        (utliste '()))
    (dolist (triplet kat-liste-avsn)
            (push (list (car triplet)
                        (cadr triplet)
                        (returner-leksikon-frekvens-mest-frekvente (cdr triplet)
                                                                   antall))
                  utliste))
    utliste))

(defun sammenlign-alle-navn-frekvens-mest-frekvente (&optional
                                                     (antall 50)
                                                     (kategoriliste '())
                                                     (ant-ord-grense-person 0))
  "Returnerer frekvensen for alle pnavn med avsn sortert etter total
   frekvens. Kan velge å analysere bare et utvalg kategorier, og kan
   utelate personer med få ord."
  (let ((utliste '()))
    (dolist (triplet (hash2list *frekvenstabeller*))
            (let ((pnavn (finn-navn (car triplet))))
              (when (and (or (null kategoriliste)
                             (find (pnavnkategori pnavn)
                                   kategoriliste :test #'string-equal))
                         (> (cadr triplet) ant-ord-grense-person))
                (push (list (format '() "~D" (car triplet))
                            (cadr triplet)
                            (returner-leksikon-frekvens-mest-frekvente (cdr triplet)
                                                                       antall))
                      utliste))))
    utliste))

(defun sammenlign-alle-navn-frekvens-mest-frekvente (&optional
                                                     (antall 50)
                                                     (kategoriliste '()))
  "Returnerer frekvensen for alle pnavn med avsn sortert etter total
   frekvens. Kan velge å analysere bare et utvalg kategorier."
  (let ((utliste '()))
    (dolist (triplet (hash2list *frekvenstabeller*))
      (when (or (null kategoriliste)
                (find (pnavnkategori (finn-navn (car triplet)))
                      kategoriliste :test #'string-equal))
        (push (list (format '() "~D" (car triplet))
                    (cadr triplet)
                    (returner-leksikon-frekvens-mest-frekvente (cdr triplet)
                                                               antall))
              utliste)))
    utliste))

(defun sammenlign-alle-navn-frekvens-ett-ord (ord)
  "Returnerer frekvensen for et ord for alle pnavn med avsn."
  (let ((utliste '()))
    (dolist (triplet (hash2list *frekvenstabeller*))
```

```
           (push (list (format '() "~D" (car triplet))
                       (cadr triplet)
                       (returner-leksikon-frekvens-ett-ord (cdr triplet)
                                                           ord))
                 utliste))
    utliste))


(defun snittlengde-ord (ordliste)
  "Tar ordliste med (ord . antall) og returnerer gjennomsnittlig
   ordlengde, samlet ordlengde og en vektor med alle ordlengdene."
  (let ((ord 0)
        (bokstaver 0)
        (lengdene (make-array 30 :initial-element 0)))
    (dolist (ordpar ordliste)
            (let ((ordlengde (length (car ordpar))))
              (setq bokstaver (+ bokstaver
                                 (* ordlengde
                                    (cdr ordpar))))
              (setq ord (+ ord (cdr ordpar)))
              (setf (svref lengdene ordlengde) (+ (svref lengdene ordlengde)
                                                  (cdr ordpar)))))
    (values (/ bokstaver ord)
            ord
            lengdene)))

(defun snittlengde-ord-alle-frekvenstabeller ()
  "Returnerer gjennomsnittlig ordlengde for alle personer med avsnitt."
  (let ((utliste '()))
    (maphash #'(lambda (x y)
                 (push (cons x (snittlengde-ord (hash2list (cdr y)))) utliste))
             *frekvenstabeller*)
    (reverse utliste)))

(defun skriv-snittlengde-ord-alle-frekvenstabeller (navneliste)
  "Skriver ut ei liste med navnenumre og gjennomsnittlig ordlengde."
  (format t "~%Ordlengede for alle navn:~%")
  (dolist (navnepar navneliste)
          (let* ((navnid (car navnepar))
                 (navn (finn-navn navnid))
                 (navnet (navnet navn))
                 (navntrunk (if (> (length navnet) 29)
                                (subseq navnet 0 28)
                                navnet))
                 (lengde (cdr navnepar)))
            (format t "~%~5,,,D ~30,,,A ~15,,,A ~,7,,,@F"
                    navnid
                    navntrunk
                    (pnavnkategori navn)
                    lengde))))

(defun lengde-signatur-ord-alle-frekvenstabeller ()
  "Returnerer gjennomsnittlig ordlengde og snittlengde-signatur for
   alle personer med avsnitt."
  (let ((utliste '()))
    (maphash #'(lambda (x y)
                 (multiple-value-bind
                  (snitt antord lengdene)
```

```
                (snittlengde-ord (hash2list (cdr y)))
                (push (list x antord snitt lengdene) utliste)))
            *frekvenstabeller*)
    (reverse utliste)))

(defun skriv-snittlengde-signatur-ord-alle-frekvenstabeller (navneliste)
  "Skriver ut ei liste med navnenumre, gjennomsnittlig % ordlengde og
   snittfordeling."
  (format t "~%Ordlengede for alle navn:~%")
  (dolist (navnepar navneliste)
          (let* ((navnid (car navnepar))
                 (navn (finn-navn navnid))
                 (navnet (navnet navn))
                 (navntrunk (if (> (length navnet) 29)
                                (subseq navnet 0 28)
                              navnet))
                 (antord (cadr navnepar))
                 (lengdesnitt (caddr navnepar))
                 (sign-vector (cadddr navnepar)))
            (format t "~%~5,,,D|~30,,,A|~15,,,A|~,7,,,@F~{|~,3,,,@F~}"
                    navnid
                    navntrunk
                    (pnavnkategori navn)
                    lengdesnitt
                    (mapcar #'(lambda (x)
                                (* 100 (/ x antord)))
                            (vect2list sign-vector))))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Naboanalyse                                                        ;
;--------------------------------------------------------------------;

(defun ord-etter-node (node)
  "Finner ordet rett etter node."
  (when (typep node 'element)
    (let* ((neste-noder (sosken-etter node))
           (ordene (hent-ord-s (finn-pcdata-fra-tre neste-noder
                                                    :ikke-noder '("pnavn" "snavn"))
                               '"1234567890-/"))
           (mamma (nodens-mamma node)))
      (if ordene
          (let ((ord (finn-foerste-ord ordene)))
            (if ord
                ord
                (if (and (typep mamma 'element)
                         (equal-case (elementnavn mamma) '"avsn"))
                    '()
                    (ord-etter-node mamma))))
          (if (and (typep mamma 'element)
                   (equal-case (elementnavn mamma) '"avsn"))
              '()
              (ord-etter-node mamma))))))

(defun ord-etter-node-s (node)
  "Finner ordet rett etter node. Skalerer bedre."
  (when (typep node 'element)
```

```
    (let* ((neste-noder (sosken-etter node))
           (nodenrpar (mapcar #'(lambda (node)
                                  (let* ((id-fra (nodeid node))
                                         (id-til (finn-siste-nodeid node)))
                                    (cons id-fra id-til)))
                              neste-noder))
           (pcdata '""))
      (dolist (par nodenrpar)
        (setq pcdata
              (concatenate 'string pcdata
                           (finn-pcdata-fra-til (car par)
                                                (cdr par)
                                                :ikke-noder '("pnavn" "snavn")))))
      (let ((ordene (hent-ord-s pcdata))
            (mamma (nodens-mamma node)))
        (if ordene
            (let ((ord (finn-foerste-ord ordene)))
              (if ord
                  ord
                (if (and (typep mamma 'element)
                         (equal-case (elementnavn mamma) '"avsn"))
                    '()
                  (ord-etter-node-s mamma))))
          (if (and (typep mamma 'element)
                   (equal-case (elementnavn mamma) '"avsn"))
              '()
            (ord-etter-node-s mamma)))))))

(defun ord-mellom-noder (node)
  "Finner ordet mellom snavnet node og neste snavn."
  (when (typep node 'element)
    (let* ((neste-noder (sosken-etter node))
           (noder-til (finn-noder-i-subtre neste-noder '"snavn")))
      (if noder-til
          (let ((ord-mellom
                 (fjern-streng-fra-liste
                  '"mdash"
                  (hent-ord (rydd-streng (finn-pcdata-fra-til
                                          (nodeid (car neste-noder))
                                          (nodeid (car noder-til)))) '"-"))))
            (if ord-mellom
                ord-mellom
                (list '"INGEN ORD")))
          (let ((mamma (nodens-mamma node)))
            (if (and (typep mamma 'element)
                     (equal-case (elementnavn mamma) '"avsn"))
                (list '"SISTE ORD")
                (ord-mellom-noder mamma)))))))

(defun pnavn-kontekst-etter (avsn)
  "Finner alle kontekster (første ord rett etter) til pnavn i avsn."
  (when (and (typep avsn 'element)
             (equal-case (elementnavn avsn) '"avsn"))
    (let ((navnnoder (finn-noder-i-subtre (list avsn) '"snavn")))
      (mapcar #'ord-etter-node navnnoder))))

(defun pnavn-kontekst-mellom (avsn)
```

```
  "Finner alle kontekster mellom to pnavn i avsn."
  (when (and (typep avsn 'element)
             (equal-case (elementnavn avsn) '"avsn"))
    (let ((navnnoder (finn-noder-i-subtre (list avsn) '"snavn")))
      (mapcar #'ord-mellom-noder navnnoder))))

(defun finn-foerste-ord (ordliste)
  "Hopper over innledende mdash."
  (cond ((null ordliste)
         '())
        ((equal-case (car ordliste) '"mdash")
         (finn-foerste-ord (cdr ordliste)))
        (t
         (car ordliste))))

(defun finn-pnavn-kontekst-for-pnavn (pnavn &optional (max-lengde -1))
  "Skriver ut alle kontekster mellom to snavn i avnsitt der pnavn er taler."
  (let ((avsnittene (pnavntaleravsn pnavn)))
    (when avsnittene
      (kutt-indre-lister (mapcar #'pnavn-kontekst-mellom avsnittene)
                         max-lengde))))

(defun finn-pnavn-kontekst-etter-snavn (pnavn)
  "Finner alle kontekster etter snavn i avsnitt tilhørende pnavn."
  (let ((frekvensliste (make-hash-table :test #'equal))
        (ordliste (flat-ut (mapcar #'pnavn-kontekst-etter
                                   (pnavntaleravsn pnavn)))))
    (dolist (ord ordliste)
      (let ((frekvens (gethash ord frekvensliste)))
        (setf (gethash ord frekvensliste)
              (if frekvens
                  (+ frekvens 1)
                1))))
    (cons (length ordliste) frekvensliste)))


(defun lag-kontekst-etter-snavn-tabeller ()
  "Lager kontekst etter snavn-tabeller for alle pnavn."
  (clrhash *kontekst-etter-snavn-tabeller*)
  (maphash
   #'(lambda (id pnavn)
       (let ((avsnitt (pnavntaleravsn pnavn)))
         (when avsnitt
           (setf (gethash id *kontekst-etter-snavn-tabeller*)
                 (finn-pnavn-kontekst-etter-snavn pnavn))
           (format t "Skrevet pnavn ~D, ~D avsnitt.~%" id (length avsnitt)))))
   *navnene*))

(defun skriv-samlet-kontekst-tabell (&optional (kategori '()))
  "Lager en samlet kontekstabell for alle avsnitt tilordnet aktuelle
   talere av oppgitt kategori (ingen parameter gir alle). Det betyr at
   'skal ikke analyseres' kuttes ut."
  (let ((leksikonet (make-hash-table :test #'equal))
        (antall 0))
    (maphash #'(lambda (id frekvenspar)
                 (when (not (= id 3)) ; Kutter ut "Skal ikke analyseres"
                   (let* ((pnavn (finn-navn id))
                          (pnavnkat (pnavnkategori pnavn)))
```

```
                    (when (or (not kategori)
                              (equal-case pnavnkat kategori))
                      (dolist (frekvpar (hash2list (cdr frekvenspar)))
                        (let* ((ord (car frekvpar))
                               (frekvens (cdr frekvpar))
                               (frekv-til-naa (gethash ord leksikonet)))
                          (setf (gethash ord leksikonet)
                                (if frekv-til-naa
                                    (+ frekvens frekv-til-naa)
                                  frekvens))))
                      (setq antall (+ antall (car frekvenspar)))
                      (format t "Lagt til navn ~D...~%" id)))))
            *kontekst-etter-snavn-tabeller*)
    (cons antall leksikonet)))

(defun lag-etter-snavn-tabeller-alle ()
  "Lager en samlet kontekstttabell etter snavn for alle pnavn."
  (setq *kontekst-etter-snavn-tabeller-alle*
        (skriv-samlet-kontekst-tabell))
  'gjort)

(defun lag-etter-snavn-tabeller-alle-sortert ()
  "Lager en samlet kontekstttabell etter snavn for alle pnavn sortert
   etter frekvens."
  (setq *kontekst-etter-snavn-tabeller-alle-sort*
        (sort (hash2list (cdr *kontekst-etter-snavn-tabeller-alle*))
              #'(lambda (x y) (> (cdr x) (cdr y)))))
  'gjort)

(defun skriv-etter-frekvens-diff-liste (diffliste)
  "Skriver ut ei diffliste som er på formen (navn antall-ord (differanser))."
  (format t "~%                      ~{~10,,,@A ~}"
          (mapcar #'(lambda (x)
                      (let ((navn (car x)))
                        (if (> (length navn) 8)
                            (subseq navn 0 9)
                          navn)))
                  diffliste))
  (format t "~%           ~10,,,D " (car *kontekst-etter-snavn-tabeller-alle*))
  (format t "~{~10,,,D ~}" (mapcar #'cadr diffliste))
  (let ((lengde (length diffliste)))
    (dolist (ordpar (caddar diffliste))
      (format t "~%~10,,,A ~10,,,D ~,7,,,@F "
              (car ordpar)
              (gethash (car ordpar) (cdr *kontekst-etter-snavn-tabeller-alle*))
              (cdr ordpar))
      (dotimes (teller lengde)
        (let* ((ordliste (caddr (elt diffliste teller)))
               (indre-ordpar (assoc (car ordpar) ordliste :test #'equal)))
          (format t "~,7,,,@F " (cdr indre-ordpar)))))))

(defun returner-etter-tabeller-frekvens-mest-frekvente (lexpar &optional (antall '()))
  "Returnerer frekvensen av leksikonet sortert etter synkende total
   frekvens."
  (let ((ord-analyse
         (if antall
             (subseq *kontekst-etter-snavn-tabeller-alle-sort* 0 (- antall 1))
```

```
          *kontekst-etter-snavn-tabeller-alle-sort*))
        (lex (cdr lexpar))
        (antord (car lexpar))
        (utliste '()))
    (mapc #'(lambda (frekvenspar)
              (let* ((ord (car frekvenspar))
                     (f (gethash ord lex))
                     (frekvens (if f f 0)))
                (push (cons ord (/ frekvens antord)) utliste)))
          ord-analyse)
    (reverse utliste)))

(defun sammenlign-alle-navn-etter-tabeller-frekvens-mest-frekvente
  (&optional (antall 50) (kategoriliste '()) (ant-ord-grense-person 0))
  "Returnerer frekvensen for alle pnavn med avsn sortert etter total
   frekvens. Kan velge å analysere bare et utvalg kategorier."
  (let ((utliste '()))
    (dolist (triplet (hash2list *kontekst-etter-snavn-tabeller*))
      (when (and (or (null kategoriliste)
                     (find (pnavnkategori (finn-navn (car triplet)))
                           kategoriliste :test #'string-equal))
                 (> (cadr triplet) ant-ord-grense-person))
        (push (list (format '() "~D" (car triplet))
                    (cadr triplet)
                    (returner-etter-tabeller-frekvens-mest-frekvente
                     (cdr triplet)
                     antall))
              utliste)))
    utliste))

(defun ordene-etter-node (node &optional (ordform '()) (kontext 10))
  "Finner ordene etter node i samme avsn. Hvis ordform er oppgitt,
   finnes kun kontekster som innledes med ordformene."
  (when (typep node 'element)
          ; For å få tak i kontekst der det er flere pnavn under
          ; samme kur, må vi gå oppover i treet her. Henter derfor
          ; alle noder fram til slutten på avsnittet.
    (let* ((avsnittsnode (finn-node-over node '"avsn"))
           (sosken-etter (sosken-rett-etter-eller-over node '"avsn"))
           (ordene (if sosken-etter
                       (hent-ord-s (finn-pcdata-fra-til
                                    (nodeid sosken-etter)
                                    (finn-siste-nodeid avsnittsnode)
                                    :ikke-noder '("pnavn" "snavn"))
                                   '"1234567890-/")
                     '())))
      (when ordene
        (let ((treff (search ordform ordene :test #'equal-case)))
          (when (or (not ordform)
                    (and treff (= 0 treff)))
            (lag-streng-av-liste (if (and kontext
                                          (< kontext (length ordene)))
                                     (subseq ordene 0 kontext)
                                   ordene)))))))))

(defun pnavn-hele-kontekst-etter (avsn &optional (ordform '()) (kontext 10))
  "Finner alle kontekster etter og ut avsnittet til pnavn i avsn. Hvis
```

```
   ordform er oppgitt, finnes kun kontekster som innledes med
   ordformene."
  (when (and (typep avsn 'element)
             (equal-case (elementnavn avsn) '"avsn"))
    (let ((utliste '()))
      (dolist (navnnode (finn-noder-i-subtre (list avsn) '"snavn"))
        (push (concatenate 'string
                           (ordene-etter-node navnnode
                                              ordform
                                              kontext)
                           (format '() " (avsn ~D)" (nodeid avsn)))
              utliste))
      utliste)))

(defun finn-pnavn-hele-kontekst-etter-snavn
  (pnavn &optional (ordform '()) (kontext 10))
  "Finner alle kontekster etter og ut avsnittet i avsnitt tilhørende
   pnavn. Hvis ordform er oppgitt, finnes kun kontekster som innledes med
   ordformen. I tilfelle bør kallende rutine filtrere bort NIL'ene"
  (let ((utliste '()))
    (dolist (avsnitt (pnavntaleravsn pnavn))
      (push (pnavn-hele-kontekst-etter avsnitt
                                       ordform
                                       kontext)
            utliste))
    (flat-ut utliste)))

(defun skriv-alle-pnavn-hele-kontekst-etter-snavn
  (&optional (ordform '()) (kontext 10))
  "Skriver ut alle etter-kontekster etter snavn. Hvis ordform er
   oppgitt, finnes kun kontekster som innledes med ordformene."
  (format t "~%Kontekst <SNAVN~{ ~A~}>~%" ordform)
  (let ((kategoriene
         (let ((liste '()))
           (push '"" liste)
           (maphash #'(lambda (x y)
                        (push y liste))
                    *tittel-kategori*)
           (sort (remove-duplicates liste :test #'equal) #'string<))))
    (dolist (kategorien kategoriene)
      (format t "~%Kategori ~A:~%" kategorien)
      (maphash
       #'(lambda (id pnavn)
           (let ((avsnitt (pnavntaleravsn pnavn)))
             (when (and avsnitt
                        (equal kategorien (pnavnkategori pnavn)))
               (let ((linjene
                      (sort
                       (filter
                        #'(lambda (x)
                            (if (and x
                                     (not (equal (subseq x 0 1) '" ")))
                                x
                              '()))
                        (finn-pnavn-hele-kontekst-etter-snavn pnavn ordform kontext))
                       #'string-lessp)))
                 (when linjene
```

```
                  (format t "~%~D ~A:" (navnid pnavn) (navnet pnavn))
                  (format t "~{~%SNAVN ~A~}~%" linjene))))))
      *navnene*))))

(defun finn-pnavn-hele-kontekst-mellom-snavn (pnavn &optional (maxlengde 99999))
  "Finner alle mellom snavn-kontekster for pnavn. Hvis maxlengde oppgitt, kun
   kontekster opp til den lengden."
  (let ((kontekster (finn-pnavn-hele-kontekst-etter-snavn pnavn '() maxlengde))
        (utliste '()))
    (dolist
     (kontekst kontekster)
     (let ((neste-sted (search '"SNAVN" kontekst)))
       (if neste-sted
           (let ((par (search '"(" kontekst)))
             (if par
                 (push (concatenate 'string
                                    (subseq kontekst 0 neste-sted)
                                    (subseq kontekst par))
                       utliste)
               (error "finn-pnavn-hele-kontekst-mellom-snavn: Ingen parantes!"))))))
    utliste))

(defun skriv-alle-pnavn-hele-kontekst-mellom-snavn (&optional (maxlengde 99999))
  "Skriver ut alle mellom-kontekster mellom snavn. Hvis maxlengde er
   oppgitt, kun kontekster opp til den lengden."
  (format t "~%Kontekster mellom snavn, lengde opp til ~D tegn.~%" maxlengde)
  (let ((kategoriene (let ((liste '()))
                       (push '"" liste)
                       (maphash #'(lambda (x y)
                                    (push y liste))
                                *tittel-kategori*)
                       (sort (remove-duplicates liste :test #'equal) #'string<))))
    (dolist (kategorien kategoriene)
      (format t "~%Kategori ~A:~%" kategorien)
      (maphash
       #'(lambda (id pnavn)
           (let ((avsnitt (pnavntaleravsn pnavn)))
             (when (and avsnitt
                        (equal kategorien (pnavnkategori pnavn)))
               (let ((linjene
                      (sort
                       (filter
                        #'(lambda (x)
                            (if (and x
                                     (not (equal (subseq x 0 1) '" ")))
                                x
                              '()))
                        (finn-pnavn-hele-kontekst-mellom-snavn pnavn (+ maxlengde 1)))
                       #'string-lessp)))
                 (when linjene
                   (format t "~%~D ~A: (~D kontekster)"
                           (navnid pnavn) (navnet pnavn) (length linjene))
                   (format t "~{~%~A~}~%" linjene))))))
       *navnene*))))

(defun skriv-alle-kategorier-hele-kontekst-mellom-snavn (&optional (maxlengde 99999))
  "Skriver ut alle mellom-kontekster mellom snavn gruppert på
```

```
   kategorier. Hvis maxlengde er oppgitt, kun kontekster opp til den
   lengden."
  (format t "~%Kontekster mellom snavn, lengde opp til ~D tegn " maxlengde)
  (let ((kategoriene (let ((liste '()))
                       (push '"" liste)
                       (maphash #'(lambda (x y)
                                    (push y liste))
                                *tittel-kategori*)
                       (sort (remove-duplicates liste :test #'equal) #'string<))))
    (dolist (kategorien kategoriene)
      (format t "~%Kategori ~A:~%" kategorien)
      (let ((utliste '()))
        (maphash
         #'(lambda (id pnavn)
             (let ((avsnitt (pnavntaleravsn pnavn)))
               (when
                   (and avsnitt
                        (equal kategorien (pnavnkategori pnavn)))
                 (let ((linjene
                        (filter
                         #'(lambda (x)
                             (if (and x
                                      (not (equal (subseq x 0 1) '" ")))
                                 x
                               '()))
                         (finn-pnavn-hele-kontekst-mellom-snavn
                          pnavn (+ maxlengde 1)))))
                   (when linjene
                     (setq utliste (append linjene utliste)))))))
         *navnene*)
        (format t " (~D kontekster)~%~%" (length utliste))
        (format t "~{~%~A~}~%" (sort utliste #'string-lessp))
        (let ((utliste-uten-avsn
               (mapcar
                #'(lambda (x)
                    (string-trim '" " (subseq x 0 (search '"(" x))))
                utliste)))
          (format
           t "~{~%~A~}~%"
           (mapcar #'(lambda (y)
                       (concatenate
                        'string (string-downcase y)
                        (format '() ": ~D"
                                (count y utliste-uten-avsn :test #'equal-case))))
                   (remove-duplicates utliste-uten-avsn :test #'equal-case)))))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; KWIC-konkordans                                                      ;
;----------------------------------------------------------------------;

(defun skriv-ordliste (nodeliste &key (ikke-noder '("pnavn" "snavn")))
  "Skriver ordliste for nodene med røtter i nodeliste basert på pcdata. Skalerer."
  (let ((nodenrpar (mapcar #'(lambda (node)
                               (let* ((id-fra (nodeid node))
                                      (id-til (finn-siste-nodeid node)))
                                 (cons id-fra id-til)))
```

```
                          nodeliste))
        (pcdata '""))
    (dolist (par nodenrpar)
            (setq pcdata (concatenate 'string pcdata
                                      (finn-pcdata-fra-til (car par)
                                                           (cdr par)
                                                           :ikke-noder ikke-noder))))
    (let ((ordliste (fjern-streng-fra-liste
                     '"mdash"
                     (hent-ord-s (string-downcase (rydd-streng pcdata))
                                 '"1234567890-/"))))
      ordliste)))

(defun skriv-KWIC-konk-sort-hkontxt (konk)
  "Skriver ut en KWIC-konkordans sortert på høyre-kontekst."
  (format
   t "~%<table>~{~%<tr><td align=right>~A</tr>~}~%</table>~%"
   (mapcar #'(lambda (x)
               (byttut
                '"<td><b>" '"["
                (byttut
                 '"</b><td>" '"]"
                 (lag-streng-av-liste x))))
           (sort konk
                 #'(lambda (k1 k2)
                     (let ((k1-treff
                            (position
                             '"[" k1
                             :test #'(lambda (x y) (eq (char x 0) (char y 0)))))
                           (k2-treff
                            (position
                             '"[" k2 :test #'(lambda (x y) (eq (char x 0) (char y 0))))))
                       (if (and k1 k2)
                           (string-lessp (lag-streng-av-liste (subseq k1 (+ k1-treff 1)))
                                         (lag-streng-av-liste (subseq k2 (+ k2-treff 1))))
                         (error "skriv-KWIC-konk-sort-hkontxt: feil liste inn")))))))

(defun skriv-KWIC-konk-sort-nodenr (konk)
  "Skriver ut en KWIC-konkordans sortert på nodenummer."
  (format t "~%<table>~{~%<tr><td align=right>~A</tr>~}~%</table>~%"
          (mapcar #'(lambda (x)
                      (byttut
                       '"<td><b>" '"["
                       (byttut
                        '"</b><td>" '"]"
                        (lag-streng-av-liste x))))
                  (sort konk #'(lambda (k1 k2)
                                 (let ((t1 (parse-integer (car (last k1))
                                                          :junk-allowed t :start 6))
                                       (t2 (parse-integer (car (last k2))
                                                          :junk-allowed t :start 6)))
                                   (if (and t1 t2)
                                       (< t1 t2)
                                     t)))))))


(defun KWIC-konk-nodeliste (nodeliste ordform
                                    &key (kontext 5) (trunk '())
```

```
                                    (ikke-noder '("pnavn" "snavn")))
  "Lager en KWIC-konkordans på ordform basert på nodene i nodeliste."
  (let ((utliste '()))
    (dolist (node nodeliste)
      (setq utliste
            (append utliste
                    (if trunk
                        (KWIC-konk-ordliste-trunk
                         node
                         (skriv-ordliste (list node)
                                         :ikke-noder ikke-noder) ordform kontext)
                      (KWIC-konk-ordliste
                       node
                       (skriv-ordliste (list node)
                                       :ikke-noder ikke-noder) ordform kontext)))))
    utliste))

(defun KWIC-konk-ordliste (node ordliste ordform &optional (kontext 5) (start 0))
  "Lager en usortert KWIC-konkordans på ordform basert på ordliste."
  (let ((treff (position ordform ordliste :test #'equal-case :start start)))
    (if treff
        (cons (append (subseq ordliste
                              (if (> treff kontext)
                                  (- treff kontext)
                                0)
                              treff)
                      (list (concatenate 'string
                                         '"["
                                         (elt ordliste treff)
                                         '"]"))
                      (subseq ordliste
                              (+ treff 1)
                              (if (< (+ treff kontext) (length ordliste))
                                  (+ treff kontext 1)
                                (length ordliste)))
                      (list (format '() "(avsn ~D)" (nodeid node))))
              (KWIC-konk-ordliste node ordliste ordform kontext (+ treff 1)))
      '())))

(defun KWIC-konk-ordliste-trunk (node ordliste ordform &optional (kontext 5) (start 0))
  "Lager en usortert KWIC-konkordans på alle ord som starter med
   ordform basert på ordliste."
  (let ((treff (position ordform
                         ordliste
                         :test #'(lambda (x y)
                                   (let ((len-x (length x))
                                         (len-y (length y)))
                                     (when (<= len-x len-y)
                                       (equal-case x (subseq y 0 len-x)))))
                         :start start)))
    (if treff
        (cons (append (subseq ordliste
                              (if (> treff kontext)
                                  (- treff kontext)
                                0)
                              treff)
                      (list (concatenate 'string
```

```
                                        '"["
                                        (elt ordliste treff)
                                        '"]"))
                        (subseq ordliste
                                (+ treff 1)
                                (if (< (+ treff kontext) (length ordliste))
                                    (+ treff kontext 1)
                                  (length ordliste)))
                        (list (format '() "(avsn ~D)" (nodeid node))))
                (KWIC-konk-ordliste-trunk node ordliste ordform kontext (+ treff 1)))
      '())))

(defun skriv-KWIC-kontekst-alle-kategorier-html (&key (trunk '()) (ikke-noder '()))
  "Skriver ut KWIC-konkordanser sortert på kategori."
  (format t "~%Ordform: ")
  (let ((ordform (read-line)))
    (format
     t "<HTML><HEAD><TITLE>Konkordans ~A</TITLE></HEAD>~%<BODY>~%<H1>Konkordans ~A</H1>"
     ordform ordform)
    (let ((kategoriene (let ((liste '()))
                         (push '"" liste)
                         (maphash #'(lambda (x y)
                                      (push y liste))
                                  *tittel-kategori*)
                         (sort (remove-duplicates liste :test #'equal) #'string<))))
      (dolist (kategorien kategoriene)
        (format t "~%<H2>Kategori ~A</H2>~%" kategorien)
        (let ((utliste '()))
          (maphash #'(lambda (id pnavn)
                       (let ((avsnitt (pnavntaleravsn pnavn)))
                         (when (and avsnitt
                                    (equal kategorien (pnavnkategori pnavn)))
                           (let ((linjene (KWIC-konk-nodeliste
                                           avsnitt ordform
                                           :trunk trunk
                                           :ikke-noder ikke-noder)))
                             (when linjene
                               (setq utliste (append linjene utliste)))))))
                   *navnene*)
          (skriv-KWIC-konk-sort-hkontxt utliste)))))
  (format t "</BODY></HTML>"))

(defun skriv-KWIC-kontekst-all-tekst-html (&key (trunk '()) (ikke-noder '()))
  "Skriver ut KWIC-konkordanser sortert på nodenummer."
  (format t "~%Ordform: ")
  (let ((ordform (read-line)))
    (format
     t "<HTML><HEAD><TITLE>Konkordans ~A</TITLE></HEAD>~%<BODY>~%<H1>Konkordans ~A</H1>"
     ordform ordform)
    (let ((utliste '()))
      (maphash #'(lambda (id pnavn)
                   (let ((avsnitt (pnavntaleravsn pnavn)))
                     (when avsnitt
                       (let ((linjene (KWIC-konk-nodeliste avsnitt ordform
                                                           :trunk trunk
                                                           :ikke-noder ikke-noder)))
                         (when linjene
```

```
                          (setq utliste (append linjene utliste)))))))
               *navnene*)
      (skriv-KWIC-konk-sort-nodenr utliste)))
  (format t "</BODY></HTML>"))

(defun avsn-nummer-kat-fordelt-person ()
  "Viser alle taleres avsnittsnumre sortert på talers kategori."
  (let ((kategoriene (let ((liste '()))
                       (push "" liste)
                       (maphash #'(lambda (x y)
                                    (push y liste))
                                *tittel-kategori*)
                       (sort (remove-duplicates liste :test #'equal) #'string<))))
    (dolist (kategorien kategoriene)
            (format t "~%Kategori ~A:~%" kategorien)
            (maphash #'(lambda (id pnavn)
                         (let ((avsnitt (pnavntaleravsn pnavn)))
                           (when (and avsnitt
                                      (equal kategorien (pnavnkategori pnavn)))
                             (format t "Avsnitt taler ~D:~{ ~D~}.~%"
                                     id (mapcar #'nodeid avsnitt)))))
                     *navnene*))))

(defun avsn-nummer-kat-sum ()
  "Viser alle avsnittsnumre samlet for hver talerkategori."
  (let ((kategoriene (let ((liste '()))
                       (push "" liste)
                       (maphash #'(lambda (x y)
                                    (push y liste))
                                *tittel-kategori*)
                       (sort (remove-duplicates liste :test #'equal) #'string<))))
    (dolist (kategorien kategoriene)
            (let ((utliste '()))
              (maphash #'(lambda (id pnavn)
                           (let ((avsnitt (pnavntaleravsn pnavn)))
                             (when (and avsnitt
                                        (equal kategorien (pnavnkategori pnavn)))
                               (setq utliste (append avsnitt utliste)))))
                       *navnene*)
              (format t "~%Kategori ~A:~{ ~D~}.~%"
                      kategorien (sort (mapcar #'nodeid utliste) #'<))))))
```

A.4 grunner

```
;;=============================================================
;; DEFINISJONER
;;-------------------------------------------------------------

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Globale variable DTD-lokale                                 ;
;-------------------------------------------------------------;

(setq *grunnid* 0) ; Numerisk variabel som teller opp hver navnnodes
                   ; unike id

(setq *grunnene* (make-hash-table)) ; Hashtabell som holder grunnene

(setq *node-grunn* (make-hash-table)) ; Hashtabell som holder pekere
                                      ; fra nodeider til begrunnelser

(setq *navn-grunn* (make-hash-table)) ; Hashtabell som holder pekere
                                      ; fra navnider til begrunnelser

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Klassedeklarasjon for grunner                               ;
;-------------------------------------------------------------;

(defclass grunn ()
  ((id :initarg :id :reader grunnid)
   (type :initarg :type :reader type)
   (aarsak :initarg :aarsak :reader aarsak)
   (ansvarlig :initarg :ansvarlig :reader ansvarlig)
   (objekter :initarg :objekter :reader grunn-objekter))
  (:documentation "Klasse for begrunnelse av koblinger som inneholder
aarsak, ansvarlig (funksjonsnavn eller bruker), og objekter, som er ei
liste av par av type og ID til objektene."))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Innlegging av nye grunner                                   ;
;-------------------------------------------------------------;


(defun ny-grunn (id type aarsak ansvarlig berorte-objekter)
  "Lager et nytt grunn-objekt."
  (let ((grunn (make-instance 'grunn
                              :id id
                              :type type
                              :aarsak aarsak
                              :ansvarlig ansvarlig
                              :objekter
                              berorte-objekter)))
    (setf (gethash id *grunnene*) grunn)
    (legg-til-pekere-til-grunn berorte-objekter grunn)))

(defun legg-til-pekere-til-grunn (objekter grunn)
  "Leser lista med par i objekter og legger inn pekere til
grunnen."
  (if (null objekter)
```

```
      '()
    (let ((obj-type (caar objekter))
          (obj-id (cdar objekter)))
      (cond ((eq obj-type 'node)
             (let ((nodeid (gethash obj-id *node-grunn*)))
               (if nodeid
                   (setf (gethash obj-id *node-grunn*)
                         (remove-duplicates
                          (concatenate 'cons nodeid (list grunn))))
                 (setf (gethash obj-id *node-grunn*) (list grunn)))))
            ((eq obj-type 'navn)
             (let ((navnid (gethash obj-id *navn-grunn*)))
               (if navnid
                   (setf (gethash obj-id *navn-grunn*)
                         (remove-duplicates
                          (concatenate 'cons navnid (list grunn))))
                 (setf (gethash obj-id *navn-grunn*) (list grunn)))))
            (t
             (error "legg-til-pekere-til-grunn: Feil grunntype.")))
      (legg-til-pekere-til-grunn (cdr objekter) grunn))))

(defun be-om-grunn (id ansvarlig berorte-objekter)
  "Ber brukeren begrunne et valg."
  (format t
   "~%~%Hva er årsaken til dette valget? Trykk bare enter for ingen begrunnelse.~%> ")
  (ny-grunn id (read-line) (concatenate 'string ansvarlig
                                        '" - brukervalg") berorte-objekter))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Utskrift av grunner                                                  ;
;----------------------------------------------------------------------;

(defun finn-grunn (id)
  "Returnerer peker til grunn-objektet med gitt id"
  (gethash id *grunnene*))

(defun skriv-grunn-koblinger (koblinger &optional (teller 1))
  "Skriver ut alle koblingene til en grunn."
  (unless (null koblinger)
    (let ((obj-type (caar koblinger))
          (obj-id (cdar koblinger)))
      (cond ((eq obj-type 'node)
             (format t "~%~D. Kobling til node ~D med tekst:~%~A" teller obj-id
                     (finn-pcdata-fra-tre (list (finn-node obj-id)))))
            ((eq obj-type 'navn)
             (format t "~%~D. Kobling til navn ~D: ~A" teller obj-id
                     (navnet (finn-navn obj-id))))
            (t
             (error "skriv-grunn-koblinger: Feil objekttype"))))
    (skriv-grunn-koblinger (cdr koblinger) (+ teller 1))))

(defun skriv-grunn (grunn)
  "Skriver ut en grunn."
  (format t "~%GRUNN ~D av type ~A" (grunnid grunn) (type grunn))
  (format t "~%     Årsak: ~A~%     Ansvarlig: ~A"
          (aarsak grunn) (ansvarlig grunn))
```

```
  (format t "~%~%Tilknyttede objekter:")
  (skriv-grunn-koblinger (grunn-objekter grunn))
  (format
   t "~%================================================================~%"))

(defun skriv-alle-grunner ()
  "Skriver ut alle grunnene."
  (maphash (lambda (x y)
             (skriv-grunn y))
           *grunnene*))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Finne grunner tilknyttet bestemte objekter                ;
;-----------------------------------------------------------;

(defun skriv-node-grunn (node)
  "Skriver ut grunner knyttet til en bestemt node."
  (let ((nodeid (if node
                    (nodeid node)
                  (progn
                    (format t "~%Nodeid: ")
                    (parse-integer (read-line) :junk-allowed t)))))
    (format t "~%Grunner med tilknyting til node ~D:~%" nodeid node)
    (mapc #'skriv-grunn (gethash nodeid *node-grunn*))))

(defun skriv-grunner-noder-med-taler-kat (kat)
  "Skriver ut alle grunner knyttet til avsnitt med taler med kategori kat."
  (let ((kategori (if kat
                      kat
                    (progn
                      (format t "~%Kategori: ")
                      (read-line))))
        (utliste '()))
    (maphash #'(lambda (id pnavn)
                 (let ((avsnitt (pnavntaleravsn pnavn)))
                   (when (and avsnitt
                              (equal kategori (pnavnkategori pnavn)))
                     (setq utliste (append avsnitt utliste)))))
             *navnene*)
    (mapc #'skriv-node-grunn utliste)))
```

A.5 hjelpefunksjoner

```
;;;*************************************************************
;;;
;;;     HJELPEFUNKSJONER
;;;
;;;=============================================================

(defun tom-strengp (streng)
  "Sjekker om variabelen som kommer inn er en tom streng."
  (declare (string streng))
  (string= streng ""))

;;; Konverteringsrutinen for romertall er hentet fra
;;; http://pleac.sourceforge.net/pleac_commonlisp.html

(defun romanchar (x)
  (declare (character x))
  (cond ((or (eq x #\m) (eq x #\M)) 1000)
        ((or (eq x #\d) (eq x #\D)) 500)
        ((or (eq x #\c) (eq x #\C)) 100)
        ((or (eq x #\l) (eq x #\L)) 50)
        ((or (eq x #\x) (eq x #\X)) 10)
        ((or (eq x #\v) (eq x #\V)) 5)
        ((or (eq x #\i) (eq x #\I)) 1)
        (t 0)))

(defun roman2arabic (x)
  (declare (character x))
  (let ((y (coerce x 'list)))
    (if (eq 1 (length y))
        (romanchar (car y))
      (if (< (romanchar (car y)) (romanchar (cadr y)))
          (- (roman2arabic (rest y)) (romanchar (car y)))
        (+ (romanchar (car y)) (roman2arabic (rest y)))))))

(defun romanp (x)
  "Sjekker om en streng syntaktisk ligner på et romertall."
  (and (stringp x)
       (or (tom-strengp x)
           (and (position (char x 0) '"iIvVxXlLcCdDmM")
                (romanp (subseq x 1))))))

(defun filter (fn lst)
  "Hentet fra Graham, Paul: ANSI Common Lisp. Upper Saddle River, NJ, 1996, s. 105"
  (let ((acc nil))
    (dolist (x lst)
      (let ((val (funcall fn x)))
        (if val (push val acc))))
    (nreverse acc)))

(defun enumerate-interval (low high)
  (declare (fixnum low high))
  (if (> low high)
      '()
      (cons low (enumerate-interval (+ low 1) high))))
```

```
(defun byttut (erstatt orig streng &optional (strengut ""))
  "Som substitute, men godtar lengre orig og erstatt enn ett tegn."
  (declare (string erstatt orig streng strengut))
  (if (tom-strengp streng)
      strengut
    (let ((pos (search orig streng)))
      (if pos
          (byttut erstatt
                  orig
                  (subseq streng (+ pos (length orig)) )
                  (concatenate 'string strengut (subseq streng 0 pos) erstatt))
        (concatenate 'string strengut streng)))))

(defun split (streng tegn)
  "Returnere ei liste av strengen splittet. Tegn kan være character eller streng."
  (declare (string streng))
  (let* ((tegnstreng (string tegn))
         (pos (search tegnstreng streng)))
    (if pos
        (append (list (string-trim (concatenate 'string '" " (string (code-char 10))
                                                (string (code-char 13)))
                                   (subseq streng 0 pos)))
                (split (subseq streng (+ pos (length tegnstreng))) tegn))
      (list (string-trim (concatenate 'string '" " (string (code-char 10))
                                      (string (code-char 13))) streng)))))

(defun lag-streng-av-liste (liste)
  "I tillegg til å endre lista til streng settes blank mellom strengene."
  (declare (cons liste))
   (let ((streng (format nil "~{~A ~}" liste)))
     (if (tom-strengp streng)
         streng
         (subseq streng 0 (- (length streng) 1)))))

(defun siste-tegn (streng)
  "Returnerer siste tegn i ei streng."
  (when (stringp streng)
    (char streng (- (length streng) 1))))

(defun hent-ord (streng &optional (tilleggstegn '"") (liste '()))
  "Henter alle samlinger av bokstaver fra streng. Basert på 8-bits
   tegnsett i 8859-serien"
  (declare (string streng tilleggstegn))
  (let ((bokstaver (concatenate 'string *bokstaver* tilleggstegn)))
    (if (position-if #'(lambda (tegn) (find tegn bokstaver))
                     streng)
        (let* ((start-ord (position-if #'(lambda (tegn) (find tegn bokstaver))
                                       streng))
               (slutt-ord (position-if-not #'(lambda (tegn) (find tegn bokstaver))
                                           streng
                                           :start start-ord)))
          (if slutt-ord
              (hent-ord (subseq streng slutt-ord)
                        tilleggstegn
                        (cons (subseq streng start-ord slutt-ord) liste))
            (hent-ord (subseq streng (length streng))
```

```
                    tilleggstegn
                    (cons (subseq streng start-ord) liste))))
      (reverse liste))))

(defun hent-ord-s (streng &optional (tilleggstegn ""))
  "Henter alle samlinger av bokstaver fra streng. Basert på 8-bits
   tegnsett i 8859-serien. Skalerer."
  (let ((bokstaver (concatenate 'string *bokstaver* tilleggstegn))
        (lengde (length streng))
        (liste '())
        (slutt-ord 0))
    (do* ((start-ord (position-if #'(lambda (tegn) (find tegn bokstaver))
                                  streng
                                  :start slutt-ord)
                     (position-if #'(lambda (tegn) (find tegn bokstaver))
                                  streng
                                  :start slutt-ord))
          (slutt-ord (position-if-not #'(lambda (tegn) (find tegn bokstaver))
                                      streng
                                      :start (if start-ord
                                                 start-ord
                                                 lengde))
                     (position-if-not #'(lambda (tegn) (find tegn bokstaver))
                                      streng
                                      :start (if start-ord
                                                 start-ord
                                                 lengde))))
         ((or (not start-ord) (not slutt-ord))
          (when start-ord
            (push (subseq streng start-ord) liste))
          (reverse liste))
      (if slutt-ord
          (push (subseq streng start-ord slutt-ord) liste)))))

(defun rydd-streng (streng)
  "Fjerner bindestrek-linjeskift og erstatter linjeskift med blank"
  (declare (string streng))
  (byttut '" " '"  "
          (substitute (code-char 32)
                      (code-char 10)
                      (byttut '""
                              (concatenate 'string '"-" (string (code-char 10)))
                              streng))))

(defun korteste-lengde-lik (s1 s2)
  "Sammenligner to strenger som en serie ord, sammenligner kun lengden til korteste."
  (declare (string s1 s2))
  (let* ((liste1 (hent-ord s1))
         (lengde1 (length liste1))
         (liste2 (hent-ord s2))
         (lengde2 (length liste2))
         (lengde (if (< lengde1 lengde2)
                     lengde1
                     lengde2)))
    (not (mismatch liste1 liste2 :test #'equal-case :end1 lengde :end2 lengde))))

(defun equal-case (s1 s2)
```

```
  "True hvis downcase av de to er like."
  (declare (string s1 s2))
  (equal (string-downcase s1)
         (string-downcase s2)))

(defun fjern-flere-blanke (streng)
  "Erstatter gjentatte blanke med en blank."
  (declare (string streng))
  (if (search '"  " streng)
      (fjern-flere-blanke (byttut '" " '"  " streng))
    streng))


(defun finn-igjen-soundex (streng liste)
  "Returnerer streng dersom strengen finnes som element i lista eller
   strengen er lik første del av et element i lista. Sammenligner opp til
   lengden på korteste streng."
  (declare (string streng) (cons liste))
  (if (null liste)
      '() ; Fant ingen like, returnerer false
    (let* ((listestreng (car liste))
           (len-s (length streng))
           (len-l (length listestreng))
           (lengde (if (< len-s len-l)
                       len-s
                     len-l)))
      (if (or (eq (soundex-kod (string-downcase (subseq streng 0 lengde)))
                  (soundex-kod (string-downcase (subseq listestreng 0 lengde))))
              (eq (soundex-kod (string-downcase (subseq streng
                                                        (- len-s lengde)
                                                        len-s)))
                  (soundex-kod (string-downcase (subseq listestreng
                                                        (- len-l lengde)
                                                        len-l)))))
          (subseq streng 0 lengde)
        (finn-igjen-soundex streng (cdr liste))))))

(defun finn-igjen (streng liste)
  "Returnerer streng dersom strengen finnes som element i lista eller
   strengen er lik første del av et element i lista. Sammenligner opp til
   lengden på korteste streng."
  (declare (string streng) (cons liste))
  (if (null liste)
      '() ; Fant ingen like, returnerer false
    (let ((listestreng (car liste)))
      (if (and (not (tom-strengp streng))
               (not (tom-strengp listestreng))
               (finn-igjen-soundex streng (hent-ord listestreng)))
          streng
        (finn-igjen streng (cdr liste))))))

(defun flat-ut (liste)
  "Flater ut lista. Tomme lister inni lista beholdes."
  (declare (cons liste))
  (cond ((null liste)
         '())
        ((consp (car liste))
         (append (flat-ut (car liste))
```

```
              (flat-ut (cdr liste))))
        (t
         (cons (car liste)
               (flat-ut (cdr liste))))))

(defun soundex-kod (streng &optional (kode ""))
  "Soundex-koder en streng og returnerer koden som et symbol. Hvis
   strengen starter med ikke-bokstaver erstattes første tegn med X, da er
   vi uansett på ville veier."
  (declare (string streng kode))
  (cond ((= (length kode) 6)
         (read-from-string kode))
        ((tom-strengp streng)
         (read-from-string
          (concatenate 'string kode
                       (make-string (- 6 (length kode)) :initial-element #\0))))
        ((tom-strengp kode)
         (let ((forste (elt streng 0)))
           (soundex-kod (subseq streng 1)
                        (make-string 1
                                     :initial-element
                                     (if (find forste *bokstaver*)
                                         forste
                                       #\X)))))
        (t
         (let* ((bokstav (elt streng 0))
                (kodet (case bokstav
                         ((#\b #\f #\p #\v) "1")
                         ((#\c #\g #\j #\k #\q #\s #\x #\z) "2")
                         ((#\d #\t) "3")
                         ((#\l) "4")
                         ((#\m #\n) "5")
                         ((#\r) "6")
                         (otherwise ""))))
           (soundex-kod (subseq streng 1) (concatenate 'string kode kodet))))))

(defun kort-ned-til (liste lengde)
  "Kapper ned til angitt lengde dersom liste er så lang."
  (declare (cons liste) (fixnum lengde))
  (if (> (length liste) lengde)
      (subseq liste 0 lengde)
    liste))

(defun tag-likp (par tag)
  "Sjekker om taggen til et parameterpar stemmer."
  (and (consp par)
       (eq (car par) tag)))

(defun fjern-streng-fra-liste (streng liste)
  "Filtrerer vekk alle forekomster av streng fra liste."
  (filter #'(lambda (x)
              (when (not (equal x streng))
                x))
          liste))

(defun hash2list (hashtabell)
  "Gjør om hashtabell til ei liste av par."
```

```
  (let ((utliste '()))
    (maphash #'(lambda (x y)
                 (push (cons x y) utliste))
             hashtabell)
    utliste))

(defun vect2list (vector)
  "Gjør om vektor til ei liste. Ikke feilkontoll."
  (let ((utliste '()))
    (dotimes (teller (- (car (array-dimensions vector)) 1))
                 (push (svref vector teller) utliste))
    (reverse utliste)))


;;;************************************************************
;;;
;;;     IMPORT - EKSPORT til lokalt lagringsformat
;;;
;;;============================================================

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Globale variable                                            ;
;-------------------------------------------------------------;

; Variabel som holder pekere under innlegging:

(setq *pekere* '())

;;; Format: ('barnliste '(...)) ... ('mammaliste '(.) ...


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Skriv ut alle data                                          ;
;-------------------------------------------------------------;

;;; SGML-tre:

;;; Dataformat sgml-tre (inneholder data som ikke brukes i vanlig
;;; arbeid) (unntatt hashtabeller med peker til objekter, som
;;; genereres på nytt):
;;; ('globvar ('filslutt var)
;;;       ('idnummer var)
;;;       ('pnavnreg-idstart var)
;;;       ('pnavnreg-idslutt var)
;;;       ('tomme-elementer var)
;;;       ('bokstaver var)
;;;       ('brukerdialog var))
;;; ('node (...))
;;;
;;; Hashtabell: *nodene* (settes i lag-element og lag-pcdata, kalt fra
;;; les-sgml-legg-inn-node

(defgeneric lagre-sgml-tre-skriv-node (node strom)
  (:documentation "Skriver ut en node. Alle pekere uttrykkes som id-numre."))

(defmethod lagre-sgml-tre-skriv-node ((node element) strom)
  (prin1 (list 'node
```

```
              (cons 'id (nodeid node))
              (cons 'mamma (if (nodens-mamma node)
                               (nodeid (nodens-mamma node))
                             '()))
              (cons 'barnliste (mapcar #'nodeid (nodebarn node)))
              (cons 'sidetall (sidetall node))
              (cons 'sidetalltype (sidetalltype node))
              (cons 'nodens-taler (if (nodens-taler node)
                                      (navnid (nodens-taler node))
                                    '()))
              (list 'element
                    (cons 'elementnavn (elementnavn node))
                    (cons 'elementattributtliste (elementattributtliste node))
                    (cons 'registerobjekt (if (registerobjekt node)
                                              (navnid (registerobjekt node))
                                            '()))))) strom)
  'element-skrevet)

(defmethod lagre-sgml-tre-skriv-node ((node pcdata) strom)
  (prin1 (list 'node
              (cons 'id (nodeid node))
              (cons 'mamma (nodeid (nodens-mamma node)))
              (cons 'barnliste (mapcar #'nodeid (nodebarn node)))
              (cons 'sidetall (sidetall node))
              (cons 'sidetalltype (sidetalltype node))
              (cons 'nodens-taler (if (nodens-taler node)
                                      (navnid (nodens-taler node))
                                    '()))
              (list 'pcdata
                    (cons 'pcdatainnhold (pcdatainnhold node)))) strom)
  'pcdata-skrevet)

(defun lagre-sgml-tre (filnavn)
  "Skriver alle sgml-tre-data."
  (with-open-file (fil filnavn :direction :output)
                  (prin1 (list 'globvar
                               (cons 'filslutt *filslutt*)
                               (cons 'idnummer *idnummer*)
                               (cons 'pnavnreg-idstart *pnavnreg-idstart*)
                               (cons 'pnavnreg-idslutt *pnavnreg-idslutt*)
                               (cons 'tomme-elementer *tomme-elementer*)
                               (cons 'bokstaver *bokstaver*)
                               (cons 'brukerdialog *brukerdialog*)) fil)
                  (dotimes (teller *idnummer*)
                           (lagre-sgml-tre-skriv-node (finn-node (+ teller 1)) fil)))
  (format t "~%Lagret sgml-treet..."))

;;; DOKUB:

;;; Dataformat dokub (inneholder data som ikke brukes i vanlig
;;; arbeid) (unntatt hashtabeller, som må genereres på nytt):
;;; ('globvar ('navnnummer var)
;;;       ('forrige-liste var)
;;;       ('alle-oppslags-pnavn var) (skriver ut id'ene)
;;;       ('foresporsler var)
;;;       ('sidetall-nodeid var) (skriver ut som liste)
;;;       ('mulig-talerliste var) (skriver ut id'ene)
```

```
;;;       ('uspesifisert-skriver var) (skriver ut id'en)
;;;       ('skal-ikke-analyseres var) (skriver ut id'en)
;;; ('navn (...)) (foreløpig finnes ikke snavn-objekter)
;;;
;;; Hashtabeller: *navnene* (legges rett inn i les-dokub-legg-inn-navn)
;;;               *pnavnoppsl* (legges inn i legg-til-i-pnavnoppsl,
;;;                             som kalles fra les-dokub-legg-inn-navn)
;;;               *tittel-kategori* (egen lagringsprosedyre)
;;;               *navn-navneformer* (egen lagringsprosedyre)

(defmethod lagre-dokub-skriv-navn ((objekt pnavn) strom)
  "Skriver ut et pnavn-objekt."
  (prin1
   (list
    'navn
    (cons 'id (navnid objekt))
    (cons 'navnet (navnet objekt))
    (cons 'navneformer (navneformer objekt))
    (cons 'reg-elementliste (mapcar #'nodeid (navnets-reg-elementer objekt)))
    (cons 'elementliste (mapcar #'nodeid (navnets-elementer objekt)))
    (cons 'sidehenv (navnets-sidehenv objekt))
    (cons 'se-henv (navnets-sehenv objekt))
    (list 'pnavn
          (cons 'kategori (pnavnkategori objekt))
          (cons 'tittel (pnavntittel objekt))
          (cons 'taleravsn (mapcar #'nodeid (pnavntaleravsn objekt)))))) strom)
  'navn-skrevet)

(defun lagre-dokub (filnavn)
  "Skriver alle dokub-data."
  (let ((sidetall-nodeid '()))
    (maphash #'(lambda (x y)
                 (push (cons x y) sidetall-nodeid))
             *sidetall-nodeid*)
    (with-open-file
     (fil filnavn :direction :output)
     (prin1
      (list
       'globvar
       (cons 'navnnummer *navnnummer*)
       (cons 'forrige-liste *forrige-liste*)
       (cons 'alle-oppslags-pnavn (mapcar #'nodeid *alle-oppslags-pnavn*))
       (cons 'foresporsler *foresporsler*)
       (cons 'sidetall-nodeid sidetall-nodeid)
       (cons 'uspesifisert-skriver
             (if *uspesifisert-skriver*
                           (navnid *uspesifisert-skriver*)
               '()))
       (cons 'skal-ikke-analyseres
             (if *skal-ikke-analyseres*
                 (navnid *skal-ikke-analyseres*)
               '()))
       (cons 'mulig-talerliste
             (mapcar #'(lambda (x)
                         (cond ((typep x 'navn)
                                (cons 'mulig-taler-navn (navnid x)))
                               ((typep x 'node)
```

```
                                (cons 'mulig-taler-node (nodeid x)))
                               (t
                                (error "lagre-dokub: Feil objekt i mulig-talerliste"))))
                     *mulig-talerliste*))) fil)
     (mapcar #'(lambda (x)
                 (lagre-dokub-skriv-navn (finn-navn x) fil))
             (enumerate-interval 1 *navnnummer*)))
    (format t "~%Lagret navnedata..."))

;;; Grunner:

;;; Dataformat grunner
;;; ('globvar ('grunnid var)
;;;           ('node-grunn var) (skrives ut som liste)
;;;           ('navn-grunn var) (skrives ut som liste)
;;;        ('grunn ('id var)
;;;                ('type var)
;;;                ('aarsak var)
;;;                ('ansvarlig var)
;;;                ('objekter var))) (dette er kun ID'er)
;;;
;;;
;;;Hashtabeller: *grunnene*

(defmethod lagre-begrunnelser-skriv-grunn ((objekt grunn) strom)
  "Skriver ut et grunn-objekt."
  (prin1 (list 'grunn
               (cons 'id (grunnid objekt))
               (cons 'type (type objekt))
               (cons 'aarsak (aarsak objekt))
               (cons 'ansvarlig (ansvarlig objekt))
               (cons 'objekter (grunn-objekter objekt))) strom)
  'grunn-skrevet)

(defun lagre-begrunnelser (filnavn)
  "Skriver alle begrunnelsesdata."
  (let ((node-grunn '())
        (navn-grunn '()))
    (maphash #'(lambda (x y)
                 (push (cons x (mapcar #'grunnid y)) node-grunn))
             *node-grunn*)
    (maphash #'(lambda (x y)
                 (push (cons x (mapcar #'grunnid y)) navn-grunn))
             *navn-grunn*)
    (with-open-file
     (fil filnavn :direction :output)
     (prin1 (list 'globvar
                  (cons 'grunnid *grunnid*)
                  (cons 'node-grunn node-grunn)
                  (cons 'navn-grunn navn-grunn)) fil)
     (dotimes (teller *grunnid*)
       (lagre-begrunnelser-skriv-grunn (finn-grunn (+ teller 1)) fil)))
    (format t "~%Lagret begrunnelser...")))

(defun lagre ()
  "Lagrer hele datastrukturen."
  (lagre-sgml-tre (concatenate 'string
```

```
                              *filtre*
                              "lagrede-data/export-sgml-tre.txt"))
  (lagre-dokub (concatenate 'string
                            *filtre*
                            "lagrede-data/export-dokub.txt"))
  (lagre-kategorier-til-fil
   (concatenate 'string
                *filtre*
                "lagrede-data/export-dokub-tit-kat.txt"))
  (lagre-navneformer-til-fil
   (concatenate 'string
                *filtre*
                "lagrede-data/export-dokub-navneformer.txt"))
  (lagre-begrunnelser
   (concatenate 'string
                *filtre*
                "lagrede-data/export-begrunnelser.txt"))
  (format t "~%Lagret datastrukturen..."))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Les inn alle data                                                    ;
;--------------------------------------------------------------------;

;;; SGML-tre:

(defun les-sgml-legg-inn-globvar (var)
  "Legger inn alle globale variable."
  (if (= (length var) 8)
      (let ((filslutt-par (elt var 1))
            (idnummer-par (elt var 2))
            (pnavnreg-idstart-par (elt var 3))
            (pnavnreg-idslutt-par (elt var 4))
            (tomme-elementer-par (elt var 5))
            (bokstaver-par (elt var 6))
            (brukerdialog-par (elt var 7)))
        (if (and filslutt-par (tag-likp filslutt-par 'filslutt))
            (setq *filslutt* (cdr filslutt-par))
          (error "les-sgml-legg-inn-globvar: Feil data - filslutt mangler."))
        (if (and idnummer-par (tag-likp idnummer-par 'idnummer))
            (setq *idnummer* (cdr idnummer-par))
          (error "les-sgml-legg-inn-globvar: Feil data - idnummer mangler."))
        (if (and pnavnreg-idstart-par (tag-likp pnavnreg-idstart-par 'pnavnreg-idstart))
            (setq *pnavnreg-idstart* (cdr pnavnreg-idstart-par))
          (error "les-sgml-legg-inn-globvar: Feil data - pnavnreg-idstart mangler."))
        (if (and pnavnreg-idslutt-par (tag-likp pnavnreg-idslutt-par 'pnavnreg-idslutt))
            (setq *pnavnreg-idslutt* (cdr pnavnreg-idslutt-par))
          (error "les-sgml-legg-inn-globvar: Feil data - pnavnreg-idslutt mangler."))
        (if (and pnavnreg-idslutt-par (tag-likp pnavnreg-idslutt-par 'pnavnreg-idslutt))
            (setq pnavnreg-idslutt (cdr pnavnreg-idslutt-par))
          (error "les-sgml-legg-inn-globvar: Feil data - pnavnreg-idslutt mangler."))
        (if (and tomme-elementer-par (tag-likp tomme-elementer-par 'tomme-elementer))
            (setq *tomme-elementer* (cdr tomme-elementer-par))
          (error "les-sgml-legg-inn-globvar: Feil data - tomme-elementer mangler."))
        (if (and bokstaver-par (tag-likp bokstaver-par 'bokstaver))
            (setq *bokstaver* (cdr bokstaver-par))
          (error "les-sgml-legg-inn-globvar: Feil data - bokstaver mangler."))
```

```
        (if (and brukerdialog-par (eq (car brukerdialog-par) 'brukerdialog))
            (setq *brukerdialog* (cdr brukerdialog-par))
          (error "les-sgml-legg-inn-globvar: Feil data - brukerdialog mangler.")))
    (error "les-sgml-legg-inn-globvar: Feil data - feil antall variable.")))

(defun les-sgml-legg-inn-node (data)
  "Oppretter en node og legger inn data. Venter med pekerne."
  (if (= (length data) 8)
      (let ((id-par (elt data 1))
            (mamma-par (elt data 2))
            (barnliste-par (elt data 3))
            (sidetall-par (elt data 4))
            (sidetalltype-par (elt data 5))
            (nodens-taler-par (elt data 6))
            (subklasse-par (elt data 7))
            (nodeobjekt '()))
        (if (and mamma-par (tag-likp mamma-par 'mamma)
                 id-par (tag-likp id-par 'id)
                 barnliste-par (tag-likp barnliste-par 'barnliste)
                 sidetall-par (tag-likp sidetall-par 'sidetall)
                 sidetalltype-par (tag-likp sidetalltype-par 'sidetalltype)
                 nodens-taler-par (tag-likp nodens-taler-par 'nodens-taler))
            (if subklasse-par
                (progn
                  (cond
                   ((tag-likp subklasse-par 'element)
                    (if (= (length subklasse-par) 4)
                        (let ((elementnavn-par (elt subklasse-par 1))
                              (elementattributtliste-par (elt subklasse-par 2))
                              (registerobjekt-par (elt subklasse-par 3)))
                          (if (and elementnavn-par
                                   (tag-likp elementnavn-par 'elementnavn)
                                   elementattributtliste-par
                                   (tag-likp elementattributtliste-par
                                             'elementattributtliste)
                                   registerobjekt-par (tag-likp registerobjekt-par
                                                                'registerobjekt))
                              (progn
                                (setq nodeobjekt
                                      (lag-element (cdr elementnavn-par)
                                                   (cdr elementattributtliste-par)
                                                   '()
                                                   (cdr id-par)))
                                (push (cons 'mammaliste
                                            (cons nodeobjekt (cdr mamma-par)))
                                      *pekere*)
                                (push (cons 'registerobjekt
                                            (cons nodeobjekt (cdr registerobjekt-par)))
                                      *pekere*))
                            (error "les-sgml-legg-inn-node: Feil data - i elementdelen")))
                      (error "les-sgml-legg-inn-node: Feil data - feil antall
          variable i elementdel.")))
                   ((tag-likp subklasse-par 'pcdata)
                    (if (= (length subklasse-par) 2)
                        (let ((pcdatainnhold-par  (elt subklasse-par 1)))
                          (if (and pcdatainnhold-par
                                   (tag-likp pcdatainnhold-par 'pcdatainnhold))
```

```
                              (progn
                                (setq nodeobjekt (lag-pcdata (cdr pcdatainnhold-par)
                                                             '()
                                                             (cdr id-par)))
                                (push (cons 'mammaliste
                                            (cons nodeobjekt (cdr mamma-par)))
                                      *pekere*))
                            (error "les-sgml-legg-inn-node: Feil data - i pcdatadelen")))
                        (error "les-sgml-legg-inn-node: Feil data - feil antall
    variable i pcdatadel.")))
                       (t
                        (error "les-sgml-legg-inn-node: Feil data - feil subklassetype.")))
                      (when (cdr barnliste-par)
                        (push (cons 'barnliste (cons nodeobjekt (cdr barnliste-par))) *pekere*))
                      (setf (sidetall nodeobjekt) (cdr sidetall-par))
                      (setf (sidetalltype nodeobjekt) (cdr sidetalltype-par))
                      (when (cdr nodens-taler-par)
                        (push (cons 'nodens-taler
                                    (cons nodeobjekt (cdr nodens-taler-par)))
                              *pekere*)))
                (error "les-sgml-legg-inn-node: Feil data - mangler subklassedel."))
            (error "les-sgml-legg-inn-node: Feil data - i hoveddelen.")))
      (error "les-sgml-legg-inn-node: Feil data - feil antall variable.")))

(defun legg-inn-peker-til-barn (node barn-idliste)
  "Legger inn peker fra en node til dens barn."
  (setf (nodebarn node)
        (mapcar #'finn-node barn-idliste)))

(defun les-aktiver-pekere ()
  "Aktiverer alle pekere i *pekere*."
  (dolist (peker *pekere* 'aktivert-pekere)
          (cond ((tag-likp peker 'barnliste)
                 (setf (nodebarn (cadr peker))
                       (mapcar #'finn-node (cddr peker))))
                ((tag-likp peker 'mammaliste)
                 (setf (nodens-mamma (cadr peker))
                       (finn-node (cddr peker))))
                ((tag-likp peker 'registerobjekt)
                 (setf (registerobjekt (cadr peker))
                       (finn-navn (cddr peker))))
                ((tag-likp peker 'nodens-taler)
                 (setf (nodens-taler (cadr peker))
                       (finn-navn (cddr peker))))
                ((tag-likp peker 'uspesifisert-skriver)
                 (setq *uspesifisert-skriver* (finn-navn (cdr peker))))
                ((tag-likp peker 'skal-ikke-analyseres)
                 (setq *skal-ikke-analyseres* (finn-navn (cdr peker))))
                ((tag-likp peker 'mulig-taler-node)
                 (push (finn-node (cdr peker)) *mulig-talerliste*))
                ((tag-likp peker 'mulig-taler-navn)
                 (push (finn-navn (cdr peker)) *mulig-talerliste*))
                (t
                 (error "les-aktiver-pekere: Ukjent pekertype."))))
  (format t "~%Aktivert pekere..."))

(defun les-sgml-tre (filnavn)
```

```
  "Leser alle sgml-tre-data."
  (with-open-file (fil filnavn :direction :input)
                  (let ((globvar (read fil)))
                    (if (and globvar
                             (eq (car globvar) 'globvar))
                        (progn
                          (les-sgml-legg-inn-globvar globvar)
                          (setq *nodene* (make-hash-table))
                          (do ((node (read fil '() 'slutt)
                                     (read fil '() 'slutt)))
                              ((eql node 'slutt))
                              (les-sgml-legg-inn-node node))
                          (setq *rot* (finn-node 1)))
                      (error "les-sgml-tre: Mangler globale variable."))))
  (format t "~%Lest sgml-tre-data..."))


;;; DOKUB:

(defun les-dokub-legg-inn-globvar (var)
  "Legger inn alle globale variable."
  (if (= (length var) 9)
      (let ((navnnummer-par (elt var 1))
            (forrige-liste-par (elt var 2))
            (alle-oppslags-pnavn-par (elt var 3))
            (foresporsler-par (elt var 4))
            (sidetall-nodeid-par (elt var 5))
            (uspesifisert-skriver-par (elt var 6))
            (skal-ikke-analyseres-par (elt var 7))
            (mulig-talerliste-par (elt var 8)))
        (if (and navnnummer-par (tag-likp navnnummer-par 'navnnummer))
            (setq *navnnummer* (cdr navnnummer-par))
          (error "les-dokub-legg-inn-globvar: Feil data - navnnummer mangler."))
        (if (and forrige-liste-par (tag-likp forrige-liste-par 'forrige-liste))
            (setq *forrige-liste* (cdr forrige-liste-par))
          (error "les-dokub-legg-inn-globvar: Feil data - forrige-liste mangler."))
        (if (and alle-oppslags-pnavn-par
                 (tag-likp alle-oppslags-pnavn-par 'alle-oppslags-pnavn))
            (setq *alle-oppslags-pnavn*
                  (mapcar #'finn-node (cdr alle-oppslags-pnavn-par)))
          (error "les-dokub-legg-inn-globvar: Feil data - alle-oppslags-pnavn mangler."))
        (if (and foresporsler-par (tag-likp foresporsler-par 'foresporsler))
            (setq *foresporsler* (cdr foresporsler-par))
          (error "les-dokub-legg-inn-globvar: Feil data - foresporsler mangler."))
        (if (and sidetall-nodeid-par (tag-likp sidetall-nodeid-par 'sidetall-nodeid))
            (progn
              (setq *sidetall-nodeid* (make-hash-table))
              (mapcar #'(lambda (x)
                          (setf (gethash (car x) *sidetall-nodeid*) (cdr x)))
                      (cdr sidetall-nodeid-par)))
          (error "les-dokub-legg-inn-globvar: Feil data - sidetall-nodeid mangler."))
        (if (and uspesifisert-skriver-par
                 (tag-likp uspesifisert-skriver-par 'uspesifisert-skriver))
            (push (cons 'uspesifisert-skriver (cdr uspesifisert-skriver-par)) *pekere*)
          (error "les-dokub-legg-inn-globvar: Feil data - uspesifisert-skriver mangler."))
        (if (and skal-ikke-analyseres-par
                 (tag-likp skal-ikke-analyseres-par 'skal-ikke-analyseres))
```

```
          (push (cons 'skal-ikke-analyseres (cdr skal-ikke-analyseres-par)) *pekere*)
        (error "les-dokub-legg-inn-globvar: Feil data - skal-ikke-analyseres mangler."))
      (if (and mulig-talerliste-par (tag-likp mulig-talerliste-par 'mulig-talerliste))
          (progn
            (setq *mulig-talerliste* '())
            (mapcar #'(lambda (x) (push x *pekere*))
                    (cdr mulig-talerliste-par)))
        (error "les-dokub-legg-inn-globvar: Feil data - mulig-talerliste mangler.")))
  (error "les-dokub-legg-inn-globvar: Feil data - feil antall variable.")))

(defun les-dokub-legg-inn-navn (data)
  "Oppretter en node og legger inn data. Legger også inn pekerne til sgml-treet."
  (if (= (length data) 9)
      (let ((id-par (elt data 1))
            (navn-par (elt data 2))
            (navneformer-par (elt data 3))
            (reg-elementliste-par (elt data 4))
            (elementliste-par (elt data 5))
            (sidehenv-par (elt data 6))
            (se-henv-par (elt data 7))
            (subklasse-par (elt data 8)))
        (if (and id-par (tag-likp id-par 'id)
                 navn-par (tag-likp navn-par 'navnet)
                 navneformer-par (tag-likp navneformer-par 'navneformer)
                 reg-elementliste-par (tag-likp reg-elementliste-par
                                                'reg-elementliste)
                 elementliste-par (tag-likp elementliste-par 'elementliste)
                 sidehenv-par (tag-likp sidehenv-par 'sidehenv)
                 se-henv-par (tag-likp se-henv-par 'se-henv))
            (let ((navnobjekt (make-instance 'pnavn
                                             :navnet (cdr navn-par)
                                             :id (cdr id-par))))
              (setf (gethash (cdr id-par) *navnene*) navnobjekt)
              (legg-til-i-pnavnoppsl (rydd-streng (cdr navn-par)) navnobjekt)
              (setf (navneformer navnobjekt) (cdr navneformer-par))
              (mapcar #'(lambda (x)
                          (navn-nytt-reg-element navnobjekt (finn-node x)))
                      (cdr reg-elementliste-par))
              (mapcar #'(lambda (x)
                          (navn-nytt-element navnobjekt (finn-node x)))
                      (cdr elementliste-par))
              (navn-ny-sidehenv navnobjekt (cdr sidehenv-par))
              (when (cdr se-henv-par)
                (navn-sett-sehenv navnobjekt (cadr se-henv-par)))
              (if subklasse-par
                  (cond
                   ((tag-likp subklasse-par 'pnavn)
                    (if (= (length subklasse-par) 4)
                        (let ((kategori-par (elt subklasse-par 1))
                              (tittel-par (elt subklasse-par 2))
                              (taleravsn-par (elt subklasse-par 3)))
                          (if (and kategori-par (tag-likp kategori-par 'kategori)
                                   tittel-par (tag-likp tittel-par 'tittel)
                                   taleravsn-par (tag-likp taleravsn-par 'taleravsn))
                              (progn
                                (pnavn-sett-kategori navnobjekt (cdr kategori-par))
                                (pnavn-sett-tittel navnobjekt (cdr tittel-par))
```

```
                                (mapcar #'(lambda (x)
                                            (pnavn-nytt-taleravsn-element navnobjekt
                                                                          (finn-node x)))
                                        (cdr taleravsn-par)))
                               (error "les-dokub-legg-inn-navn: Feil data i pnavndelen.")))
                          (error "les-dokub-legg-inn-navn: Feil antall variable i pnavndelen.")))
                      (t
                       (error "les-dokub-legg-inn-navn: Feil subklassetype.")))
                    (error "les-dokub-legg-inn-navn: Mangler subklasse.")))
              (error "les-dokub-legg-inn-navn: Feil i taggede data.")))
        (error "les-dokub-legg-inn-navn: Feil antall variable.")))

(defun les-dokub (filnavn)
  "Leser alle dokub-data."
  (with-open-file (fil filnavn :direction :input)
                  (let ((globvar (read fil)))
                    (if (and globvar
                             (eq (car globvar) 'globvar))
                        (progn
                          (les-dokub-legg-inn-globvar globvar)
                          (setq *navnene* (make-hash-table))
                          (setq *pnavnoppsl* (make-hash-table :test #'equal))
                          (do ((navn (read fil '() 'slutt)
                                     (read fil '() 'slutt)))
                              ((eql navn 'slutt))
                              (les-dokub-legg-inn-navn navn)))
                        (error "les-dokub: Mangler globale variable."))))
  (format t "~%Lest dokub-data..."))


;;; GRUNNER:

; node-grunn og navn-grunn lages i legg-til-pekere-til-grunn som
; kalles fra ny-grunn. De legges derfor ikke inn fra de globale
; variable. Det er jo snakk om en slags caching.

(defun les-grunner-legg-inn-globvar (var)
  "Legger inn alle globale variable."
  (if (= (length var) 4)
      (let ((grunnid-par (elt var 1))
            (node-grunn-par (elt var 2))
            (navn-grunn-par (elt var 3)))
        (if (and grunnid-par (tag-likp grunnid-par 'grunnid))
            (setq *grunnid* (cdr grunnid-par))
          (error "les-grunner-legg-inn-globvar: Feil data - grunnid mangler.")))
    (error "les-grunner-legg-inn-globvar: Feil data - feil antall variable.")))

(defun les-grunner-legg-inn-grunn (data)
  "Oppretter en node og legger inn data. Legger også inn pekerne til sgml-treet."
  (if (= (length data) 6)
      (let ((id-par (elt data 1))
            (type-par (elt data 2))
            (aarsak-par (elt data 3))
            (ansvarlig-par (elt data 4))
            (objekter-par (elt data 5)))
        (if (and id-par (tag-likp id-par 'id)
                 type-par (tag-likp type-par 'type)
```

```
                aarsak-par (tag-likp aarsak-par 'aarsak)
                ansvarlig-par (tag-likp ansvarlig-par 'ansvarlig)
                objekter-par (tag-likp objekter-par 'objekter))
           (ny-grunn (cdr id-par)
                     (cdr type-par)
                     (cdr aarsak-par)
                     (cdr ansvarlig-par)
                     (cdr objekter-par))
         (error "les-grunner-legg-inn-grunn: Feil i taggede data.")))
    (error "les-grunner-legg-inn-grunn: Feil antall variable.")))

(defun les-grunner (filnavn)
  "Leser alle grunn-data."
  (with-open-file (fil filnavn :direction :input)
                  (let ((globvar (read fil)))
                    (if (and globvar
                             (eq (car globvar) 'globvar))
                        (progn
                          (les-grunner-legg-inn-globvar globvar)
                          (setq *node-grunn* (make-hash-table))
                          (setq *navn-grunn* (make-hash-table))
                          (do ((grunn (read fil '() 'slutt)
                                      (read fil '() 'slutt)))
                              ((eql grunn 'slutt))
                              (les-grunner-legg-inn-grunn grunn)))
                      (error "les-grunner: Mangler globale variable."))))
  (format t "~%Lest grunn-data..."))

(defun les ()
  "Leser inn alle data til programmet."
  (les-sgml-tre (concatenate 'string
                             *filtre*
                             "lagrede-data/export-sgml-tre.txt"))
  (hent-kategorier-fra-fil  (concatenate 'string
                                         *filtre*
                                         "lagrede-data/export-dokub-tit-kat.txt"))
  (hent-navneformer-fra-fil  (concatenate 'string
                                          *filtre*
                                          "lagrede-data/export-dokub-navneformer.txt"))
  (les-dokub  (concatenate 'string
                           *filtre*
                           "lagrede-data/export-dokub.txt"))
  (les-aktiver-pekere)
  (les-grunner  (concatenate 'string
                             *filtre*
                             "lagrede-data/export-begrunnelser.txt")))
```

Tillegg B

Kjøringseksempel

Dette kjøringseksempelet viser en kjøring av programmet ved bruk av funksjenene i menysystemet. Kommentarer er skrevet med kursiv, mens det som ble skrevet inn av brukeren er understreket.

Det det står "Hvorfor det?", tilbys brukeren å skrive inn en begrunnelse for det valget som nettopp er gjort.

B.1 Oppstart

```
[3]> (load parse-dokub.lsp")
;; Loading file parse-dokub.lsp ...
;;  Loading file sgml-tre-1.3.0.lsp ...
;;  Loading of file sgml-tre-1.3.0.lsp is finished.
;;  Loading file grunner-1.0.0.lsp ...
;;  Loading of file grunner-1.0.0.lsp is finished.
;;  Loading file hjelpefunk-1.1.0.lsp ...
;;  Loading of file hjelpefunk-1.1.0.lsp is finished.
;;  Loading file analyser-1.0.0.lsp ...
;;  Loading of file analyser-1.0.0.lsp is finished.
;; Loading of file parse-dokub.lsp is finished.
T
[4]> (kjoer)
HOVEDMENY

A. Import/eksport av SGML
B. Personregister
C. Kobling taler-avsnitt
D. Frekvensanalyse
E. Naboanalyse
F. Vis tekst
G. Grunner
H. Lagring
Q. Avslutt
```

B.2 SGML-data

B.2.1 Les tekstfila

```
Valg: a

DELMENY: Import/eksport av SGML

1. Les inn SGML-tekstfila (innhold: 1 noder).
2. Les inn SGML-personnavn-registerfila (innhold: 0 noder).
3. Sett inn sidetall og nodeid på rett sted (antall sidetall lagt inn: 0).
4. Skriv ut objekttreet som SGML.
Q. Ut

Valg: 1

Antall noder: 10000
[...]
Antall noder: 120000
Sum antall noder: 129524
```

B.2.2 Les registerfila

```
Valg: 2

Antall noder: 130000
Sum antall noder: 134272

Satt inn oppslagselementene i treet...
```

B.2.3 Legg inn sidetall og id'er som attributter

```
Valg: 3

Lagt inn sidetall på nodene...
Lagt inn id som attributter...
```

B.2.4 Eksporter SGML-fil

```
Valg: 4
```

Dette valget skriver ut ei SGML-fil som er identisk med innfila, med to unntak: Nodenes `id`*-verdi er lagt inn som attributt på alle starttagger, og tekniske tagger (se fotnote 2 på side 30) er fjernet.*

B.3 Personregister

```
Valg: b
```

```
DELMENY: Personregister

1. Lag nytt navneregister (innhold: 0 navneobjekter).
2. Koble navneregister til SGML-tekstfila.
3. Vis personnavnobjektene.
4. Vis personkategoriene.
5. Skru på interaksjon.
6. Nullstill antall interaksjoner (0 til nå).
7. Vis alle taleres avsnittsnumre sortert på talers kategori.
8. Vis alle avsnittsnumre samlet for hver talerkategori.
Q. Ut

Valg: 5

DELMENY: Personregister

1. Lag nytt navneregister (innhold: 0 navneobjekter).
2. Koble navneregister til SGML-tekstfila.
3. Vis personnavnobjektene.
4. Vis personkategoriene.
5. Skru av interaksjon.
6. Nullstill antall interaksjoner (0 til nå).
7. Vis alle taleres avsnittsnumre sortert på talers kategori.
8. Vis alle avsnittsnumre samlet for hver talerkategori.
Q. Ut
```

Siden interaksjon nå er skrudd på, spørres brukeren når systemet er i tvil om hvilke former som skal legges inn.

B.3.1 Lag navneregister

```
Valg: 1

Trenger opplysninger fra bruker: En persons tittel.
Årsak: Finner ikke tittelen i tittellista.

Hele oppslagsteksten:

---> Aamund Jonsen Lappe, lapp i Jo-
kasjarf 389. <---

Alternativ 1: <lapp>
Alternativ 2: <i>
Alternativ 3: <Jokasjarf>
Alternativ 4: <.>

Skriv inn alternativ nummer eller ny tekst hvis ingen passer: 1

Ønsker du å legg inn <lapp> som ny tittel i titteloversikten?

Hvis ja: Velg en kategori titlen skal knyttes til fra lista,
```

```
skriv inn en annen hvis ingen passer,
eller bare <enter> hvis du ikke har noen kategori.

Hvis nei: Skriv inn x.

Skriv inn alternativ nummer eller ny tekst hvis ingen passer: reindriftssame

Hvorfor det?

[...]

Trenger opplysninger fra bruker: En persons tittel.
Årsak: Finner ikke tittelen i tittellista.

Hele oppslagsteksten:

---> Aanud (Aanet, Aanod, Anud) An-
dersen, sjøfinn i Repparfjord,
lagr.mann i Hammerfest pgld 241
f.  <---

Alternativ 1: <sjøfinn>
Alternativ 2: <i>
Alternativ 3: <Repparfjord>
Alternativ 4: <lagr.mann>
Alternativ 5: <i>
Alternativ 6: <Hammerfest>
Alternativ 7: <pgld>
Alternativ 8: <f.>

Skriv inn alternativ nummer eller ny tekst hvis ingen passer: 1

Ønsker du å legg inn <sjøfinn> som ny tittel i titteloversikten?

Hvis ja: Velg en kategori titlen skal knyttes til fra lista,
skriv inn en annen hvis ingen passer,
eller bare <enter> hvis du ikke har noen kategori.

Hvis nei: Skriv inn x.

Alternativ 1: <reindriftssame>

Skriv inn alternativ nummer eller ny tekst hvis ingen passer: sjøsame

Hvorfor det?

[...]

Trenger opplysninger fra bruker: En persons tittel.
Årsak: Finner ikke tittelen i tittellista.
```

```
Hele oppslagsteksten:

---> &mdash;
Guttormsen, fjellap i Lyngen-
fjord 386, 390.  <---

Alternativ 1: <fjellap>
Alternativ 2: <i>
Alternativ 3: <Lyngenfjord>
Alternativ 4: <.>

Skriv inn alternativ nummer eller ny tekst hvis ingen passer: 1

Ønsker du å legg inn <fjellap> som ny tittel i titteloversikten?

Hvis ja: Velg en kategori titlen skal knyttes til fra lista,
skriv inn en annen hvis ingen passer,
eller bare <enter> hvis du ikke har noen kategori.

Hvis nei: Skriv inn x.

Alternativ 1: <adel>
Alternativ 2: <arbeider>
Alternativ 3: <bonde>
Alternativ 4: <embedsmann>
Alternativ 5: <fastboende>
Alternativ 6: <funksjonær>
Alternativ 7: <kirke>
Alternativ 8: <kvæn>
Alternativ 9: <lagrett>
Alternativ 10: <offiser>
Alternativ 11: <politi>
Alternativ 12: <reindriftssame>
Alternativ 13: <sjøsame>
Alternativ 14: <skole>

Skriv inn alternativ nummer eller ny tekst hvis ingen passer: 12

Hvorfor det?

Trenger opplysninger fra bruker: En persons tittel.
Årsak: Finner ikke tittelen i tittellista.

Hele oppslagsteksten:

---> &mdash;
Mathiesen (Mat(t)hisen), finne-
lensmann i Varanger 328, 434 f.
 <---
```

```
Alternativ 1: <finnelensmann>
Alternativ 2: <i>
Alternativ 3: <Varanger>
Alternativ 4: <f.>

Skriv inn alternativ nummer eller ny tekst hvis ingen passer: 1

Ønsker du å legg inn <finnelensmann> som ny tittel i titteloversikten?

Hvis ja: Velg en kategori titlen skal knyttes til fra lista,
skriv inn en annen hvis ingen passer,
eller bare <enter> hvis du ikke har noen kategori.

Hvis nei: Skriv inn x.

Alternativ 1: <adel>
Alternativ 2: <arbeider>
Alternativ 3: <bonde>
Alternativ 4: <embedsmann>
Alternativ 5: <fastboende>
Alternativ 6: <funksjonær>
Alternativ 7: <kirke>
Alternativ 8: <kvæn>
Alternativ 9: <lagrett>
Alternativ 10: <offiser>
Alternativ 11: <politi>
Alternativ 12: <reindriftssame>
Alternativ 13: <sjøsame>
Alternativ 14: <skole>

Skriv inn alternativ nummer eller ny tekst hvis ingen passer: 11

Hvorfor det?

[...]

Laget navneregister...
```

B.3.2 Koble navneregister til SGML-fila

```
Valg: 2

Trenger opplysninger fra bruker: En persons navneformer.
Årsak: Navnet inneholder paranteser.
Opprinnelig navneform: Winkel(-)Zacharias
Alternativ 1: <Winkel-Zacharias>
Alternativ 2: <WinkelZacharias>
Alternativ 3: <Winkel Zacharias>
Alternativ 4: <Winkel(-)Zacharias>

Skriv inn numrene på de alternativene du vil ha med skilt med blank.
```

```
Ønskede alternativer: 1 3

Godtatte former:

Alternativ 1: <Winkel-Zacharias>
Alternativ 2: <Winkel Zacharias>

Hvilke andre navneformer bør være med? Avslutt med x.

> x

Hvorfor det?

[...]

Trenger opplysninger fra bruker: En persons navneformer.
Årsak: Inneholder komma
Opprinnelig navneform: Westen, Thomas von
Alternativ 1: <Thomas von Westen>

Skriv inn numrene på de alternativene du vil ha med skilt med blank.

Ønskede alternativer: 1

Godtatte former:

Alternativ 1: <Thomas von Westen>

Hvilke andre navneformer bør være med? Avslutt med x.

> x

Hvorfor det?

[...]

Trenger opplysninger fra bruker: En persons navneformer.
Årsak: Navnet inneholder paranteser.
Opprinnelig navneform: Tornensis (Tornexis), Anders
Alternativ 1: <Tornensis Tornexis, Anders>
Alternativ 2: <Tornensis , Anders>
Alternativ 3: <Tornensis , Anders>
Alternativ 4: <Tornexis, Anders>
Alternativ 5: <Tornensis (Tornexis), Anders>

Skriv inn numrene på de alternativene du vil ha med skilt med blank.

Ønskede alternativer: 4

Godtatte former:

Alternativ 1: <Tornexis, Anders>

Hvilke andre navneformer bør være med? Avslutt med x.
```

```
> Tornensis, Anders
> Anders Tornensis
> Anders Tornexis
> x

Hvorfor det?

[...]

Trenger opplysninger fra bruker: En persons navneformer.
Årsak: Navnet inneholder paranteser.
Opprinnelig navneform: Thomas (Thomes, Tomes) Tomesen (Brede-Thomes)
Alternativ 1: <Thomas Thomes, Tomes Tomesen Brede-Thomes>
Alternativ 2: <Thomas Tomesen>
Alternativ 3: <Thomas Tomesen>
Alternativ 4: <Thomes Tomesen>
Alternativ 5: <Tomes Tomesen>
Alternativ 6: <Thomas Brede>
Alternativ 7: <Thomas Thomes>
Alternativ 8: <Thomas (Thomes, Tomes) Tomesen (Brede-Thomes)>

Skriv inn numrene på de alternativene du vil ha med skilt med blank.

Ønskede alternativer: 2 4 5

Godtatte former:

Alternativ 1: <Thomas Tomesen>
Alternativ 2: <Thomes Tomesen>
Alternativ 3: <Tomes Tomesen>

Hvilke andre navneformer bør være med? Avslutt med x.

> Brede-Thomes
> x

Hvorfor det?

Trenger opplysninger fra bruker: En persons navneformer.
Årsak: Navnet inneholder paranteser.
Opprinnelig navneform: Thomas (Thomes, Tomes) Siursen Nørtemand
Alternativ 1: <Thomas Thomes, Tomes Siursen Nørtemand>
Alternativ 2: <Thomas Siursen Nørtemand>
Alternativ 3: <Thomas Siursen Nørtemand>
Alternativ 4: <Thomes Siursen Nørtemand>
Alternativ 5: <Tomes Siursen Nørtemand>
Alternativ 6: <Thomas (Thomes, Tomes) Siursen Nørtemand>

Skriv inn numrene på de alternativene du vil ha med skilt med blank.

Ønskede alternativer: 2 4 5
```

```
Godtatte former:

Alternativ 1: <Thomas Siursen Nørtemand>
Alternativ 2: <Thomes Siursen Nørtemand>
Alternativ 3: <Tomes Siursen Nørtemand>

Hvilke andre navneformer bør være med? Avslutt med x.

> x

Hvorfor det?

[...]

Trenger opplysninger fra bruker: En persons navneformer.
Årsak: Navnet inneholder paranteser.
Opprinnelig navneform: Rasmus Sø(f)rensen Høgh
Alternativ 1: <Rasmus Søfrensen Høgh>
Alternativ 2: <Rasmus Sørensen Høgh>
Alternativ 3: <Rasmus Sø rensen Høgh>
Alternativ 4: <Rasmus frensen Høgh>
Alternativ 5: <Rasmus Sø(f)rensen Høgh>

Skriv inn numrene på de alternativene du vil ha med skilt med blank.

Ønskede alternativer: 1 2

Godtatte former:

Alternativ 1: <Rasmus Søfrensen Høgh>
Alternativ 2: <Rasmus Sørensen Høgh>

Hvilke andre navneformer bør være med? Avslutt med x.

> x

Hvorfor det?

[...]

Trenger opplysninger fra bruker: En persons navneformer.
Årsak: Navnet inneholder paranteser.
Opprinnelig navneform: Jon (Joen) Nilsen Askepeis (Akke-)
Alternativ 1: <Jon Joen Nilsen Askepeis Akke->
Alternativ 2: <Jon Nilsen Askepeis>
Alternativ 3: <Jon Nilsen Askepeis>
Alternativ 4: <Joen Nilsen Askepeis>
Alternativ 5: <Jon Nilsen Akke>
Alternativ 6: <Jon (Joen) Nilsen Askepeis (Akke-)>

Skriv inn numrene på de alternativene du vil ha med skilt med blank.
```

```
Ønskede alternativer: 2 4

Godtatte former:

Alternativ 1: <Jon Nilsen Askepeis>
Alternativ 2: <Joen Nilsen Askepeis>

Hvilke andre navneformer bør være med? Avslutt med x.

> Jon Nilsen Akkepeis
> Joen Nilsen Akkepeis
> x

Hvorfor det?

[...]

Trenger opplysninger fra bruker: En persons navneformer.
Årsak: Navnet inneholder paranteser.
Opprinnelig navneform: Hendrik (Hendrich) Mathiesen (Mat(t)hisen)
Alternativ 1: <Hendrik Hendrich Mathiesen Matthisen>
Alternativ 2: <Hendrik Mathiesen hisen)>
Alternativ 3: <Hendrik Mathiesen hisen)>
Alternativ 4: <Hendrich Mathiesen hisen)>
Alternativ 5: <Hendrik Mathisen)>
Alternativ 6: <Hendrik thisen)>
Alternativ 7: <Hendrik Mathiesen t>
Alternativ 8: <Hendrik Mathiesen hisen>
Alternativ 9: <Hendrik (Hendrich) Mathiesen (Mat(t)hisen)>

Skriv inn numrene på de alternativene du vil ha med skilt med blank.

Ønskede alternativer:

Godtatte former:

Hvilke andre navneformer bør være med? Avslutt med x.

> Hendrik Mathiesen
> Hendrich Mathiesen
> Hendrik Mathisen
> Hendrik Matthisen
> Hendrich Mathisen
> Hendrich Matthisen
> x

Hvorfor det?

[...]

Trenger opplysninger fra bruker: En persons navneformer.
```

```
Årsak: Navnet inneholder paranteser.
Opprinnelig navneform: Aanud (Aanet, Aanod, Anud) Andersen
Alternativ 1: <Aanud Aanet, Aanod, Anud Andersen>
Alternativ 2: <Aanud Andersen>
Alternativ 3: <Aanud Andersen>
Alternativ 4: <Aanet Andersen>
Alternativ 5: <Aanod Andersen>
Alternativ 6: <Anud Andersen>
Alternativ 7: <Aanud (Aanet, Aanod, Anud) Andersen>

Skriv inn numrene på de alternativene du vil ha med skilt med blank.

Ønskede alternativer: 2 4 5 6

Godtatte former:

Alternativ 1: <Aanud Andersen>
Alternativ 2: <Aanet Andersen>
Alternativ 3: <Aanod Andersen>
Alternativ 4: <Anud Andersen>

Hvilke andre navneformer bør være med? Avslutt med x.

> x

Hvorfor det?

Satt inn navneformene på alle navn...
Satt inn pekere fra pnavnreg til pnavnnoder i teksten...
```

B.3.3 Vis personobjekter

Før dette kjøres, er en komplett datastruktur med tilordnede avsnitt lastet inn, jfr. avsnitt B.4. Kun det første personobjektet som kommer opp vises i dette utdraget, og ikke alle data for dette objektet. Men alle typer data er representert.

```
Valg: 3

Navn 448: Zacharias Olsen
Tittel: fjellfinn
Kategori: reindriftssame
Sidehenvisninger: 139; 147; 154; 156; 158 f; 161;
Navneformer: <Zacharias Olsen>

Følgende registerelementer:

Element med id 134717:
<pnavnoppsl ID="134717"><pnavn ID="134260">&mdash; Olsen</pnavn>, fjellfinn i
<snavn ID="134263">Snåsa pgld</snavn> <henv ID="134266">139, 147, 154, 156,
158 f. 161</henv>.</pnavnoppsl>

Følgende andre elementer:
```

```
Element med id 42021: <PNAVN ID="42021">Zacharias Olsen</PNAVN>

Element med id 44031: <PNAVN ID="44031">Zacharias Olsen</PNAVN>

Element med id 44172: <PNAVN ID="44172">Zacharias
Olsen</PNAVN>

Element med id 45862: <PNAVN ID="45862">Zacharias Olsen</PNAVN>

Element med id 46252: <PNAVN ID="46252">Zacharias Olsen</PNAVN>

Element med id 47006: <PNAVN ID="47006">Zacharias Olsen</PNAVN>

Element med id 47210: <PNAVN ID="47210">Zacharias Olsen</PNAVN>

Element med id 47214: <PNAVN ID="47214">Zacharias</PNAVN>

Element med id 47487: <PNAVN ID="47487">Zacharias Olsen</PNAVN>

Element med id 48054: <PNAVN ID="48054">Zacharias
Olsen</PNAVN>

Taler på følgende avsnitt:  46367; 46419; 46478; 46530; 46580;

Begrunnelser knyttet til navnet:

GRUNN 827 av type Innførsel i tittelregister
     Årsak: Fant igjen første ord etter pnavn i tittel-kategoriregister.
     Ansvarlig: parse-dokub: finn-tittel

Tilknyttede objekter:
1. Kobling til navn 448: Zacharias Olsen
2. Kobling til node 134717 med tekst:
&mdash; Olsen, fjellfinn i
Snåsa pgld 139, 147, 154, 156,
158 f. 161.
============================================================

GRUNN 828 av type Innførsel i navneregisteret
     Årsak: Første pnavn-node (objekt 1) i oppslagsnoden (objekt 2) lagt inn
som objekt i navneregisteret (objekt 3).
     Ansvarlig: parse-dokub: lag-navnereg-innforsel

Tilknyttede objekter:
1. Kobling til node 134260 med tekst:
&mdash; Olsen
2. Kobling til node 134717 med tekst:
&mdash; Olsen, fjellfinn i
Snåsa pgld 139, 147, 154, 156,
158 f. 161.
3. Kobling til navn 448: Zacharias Olsen
============================================================

GRUNN 829 av type Navneformer på et navneobjekt
     Årsak: Navnet (objekt 1) finnes i *navn-navneformer*,
så navneformene legges inn derfra.
     Ansvarlig: parse-dokub: legg-inn-navneformer-paa-ett
```

```
Tilknyttede objekter:
1. Kobling til navn 448: Zacharias Olsen
================================================================

GRUNN 1277 av type Kobling pnavn-node.
     Årsak: Soundex-kodingen av innholdet i noden (objekt 1)
er gjenfunnet i soundex-koding av en av navneformene i
pnavn-objektet (objekt 2). Sidene man leter på er hentet
fra sideinnførsler på objekt 2 etter vanlige regler.
     Ansvarlig: parse-dokub: legg-pnavn-paa-pnavnreg

Tilknyttede objekter:
1. Kobling til node 42021 med tekst:
Zacharias Olsen
2. Kobling til navn 448: Zacharias Olsen
================================================================

GRUNN 1278 av type Kobling pnavn-node.
     Årsak: Soundex-kodingen av innholdet i noden (objekt 1)
er gjenfunnet i soundex-koding av en av navneformene i
pnavn-objektet (objekt 2). Sidene man leter på er hentet
fra sideinnførsler på objekt 2 etter vanlige regler.
     Ansvarlig: parse-dokub: legg-pnavn-paa-pnavnreg

Tilknyttede objekter:
1. Kobling til node 44031 med tekst:
Zacharias Olsen
2. Kobling til navn 448: Zacharias Olsen
================================================================

[...]

GRUNN 3171 av type Taler til avnsitt
     Årsak: Taler til et avnsitt (objekt 1) satt til en node (objekt 2)
som er knyttet til et navnobjekt (objekt 3)
Tolket av Erich Helset.
     Ansvarlig: parse-dokub: velg-taler-til-avsnitt

Tilknyttede objekter:
1. Kobling til node 46367 med tekst:
Til det 2det 3die Og 4de Spørsmaal: Svarer hand det Samme Som 1te Viidne af Lap
Finnerne, dette tilleggendes ved det 4de Spørs: at hand vel i vaar har indtaget Bree Thomes,
men vil have ham derfra igien til næste Viinter, uden saa er, at de andre hands Søskinde vil
tage ham til Sig &mdash;
2. Kobling til node 46252 med tekst:
Zacharias Olsen
3. Kobling til navn 448: Zacharias Olsen
================================================================

GRUNN 3172 av type Taler til avsnitt
     Årsak: Taler til et avnsitt (objekt 1) satt til et navnobjekt (objekt 2)
Tolket av Erich Helset.
     Ansvarlig: parse-dokub: velg-taler-til-avsnitt

Tilknyttede objekter:
1. Kobling til node 46419 med tekst:
```

```
Til 11te Spørsmaal: Svarer: De Lande-Mærcker hand har hørt og veed af at Siige ere
fra Penningkiesen at reigne &mdash;
Avindsbæchen
Saxolaa-Vara &mdash;
GiepsKiach &mdash;
Rautekie-vara &mdash;
Warsands-vara &mdash;
2. Kobling til navn 448: Zacharias Olsen
============================================================

[...]
```

B.3.4 Vis personkategoriene

```
Valg: 4

Kategori: adel

Inneholder titlene: <Geheime Conferense Raad> <dronning> <konge> <prins>
=======================================

Kategori: arbeider

Inneholder titlene: <gruvearbeider>
=======================================

Kategori: bonde

Inneholder titlene: <plassmann> <nybygger> <godseier> <husmann>
<gardsarbeider> <bumann> <bonde>
=======================================

[...]
```

B.3.5 Vis avsnittsnumre

Valg 7 og 8 viser fram numrene på avsnitt tilordnet ulike talere. Kun 8 vises her.

```
Valg: 8

[...]
Kategori arbeider: 6278 6289 6379 6455 6519 [...]

Kategori bonde: 6995 7010 7055 7075 7120 7171 7181 [...]
```

B.4 Kobling taler-avsnitt

Før de neste rutinene kjøres lastes en versjon av datastrukturen med alle koblinger fra hovedmeny b (jfr. tillegg B.3), men uten noen talere tilordnet avsnitt.

```
Valg: c
```

```
DELMENY: Kobling taler-avsnitt

1. Sett inn taler på avsnitt som ikke har taler fra før.
2. Vis alle avsnittene med taler.
3. Endre taler på et avsnitt.
4. Vis statistikk for alle talere.
Q. Ut
```

B.4.1 Sett inn taler

```
Valg: 1

Kutt ut avsnittene til og med (blank for å starte med starten): 5960

WIDNERS EXAMINATION
OVER
GRENDSERNE IMELLEM NORGE NORDENFIELDS OG SVERRIGE
ANNO &mdash; 1742
PAA VINTERFØRET 1 VOLUMEN.
[node 5965, side 1]

Alternativ 1: <Uspesifisert skriver>
Alternativ 2: <Skal ikke analyseres>

Skriv nummer for forteller, blank for å søke (x avslutter innlegging): 1

Hvorfor det?

GRÆNDSE COMMISSIONS PROTOKOLL
Over De Tagne Viidner
Begyndt Paa Røraas Dend 16de Apriilis 1742ve Continueret d: 17de ibdm. paa
Koyen ved Ferragen dend 18 og 19de ibm, i Bræchen d: 21de Apriil:
I GULDALS Fogderie.
[node 5983, side 1]

Alternativ 1: <Uspesifisert skriver>
Alternativ 2: <Skal ikke analyseres>

Skriv nummer for forteller, blank for å søke (x avslutter innlegging): 1

Hvorfor det?

Efter deris Kongl: Maysts allernaadigste befalning til Major Peter Schnitler av 16
Martj Sistleeden og der paa Grundede Jnstrux fra Hr. Oberste og Kongelige grændze
Comissaire Romling til bemelte Major Schnitler af 31: ejusdem, hvilcke begge Documenter
bielegges under Lit: a: og B: har Major Schnitler begivet sig paa Reisen fra Trondhiem til
Grændzerne d: 7 Apriil nest efter og efter foregangne Reqvisition til vedckommende øfrig-
hed og betiendtere beckyndt paa Røraas først at sætte Rætten dend 16 Apriilis Nest efter.
[node 6024, side 1]

Alternativ 1: <Uspesifisert skriver>
Alternativ 2: <Peter Schnitler>
Alternativ 3: <Romling>
Alternativ 4: <Schnitler>
Alternativ 5: <Schnitler>
Alternativ 6: <Skal ikke analyseres>
```

```
Skriv nummer for forteller, blank for å søke (x avslutter innlegging): 1

Hvorfor det?

Soerenskriiveren i Guldahlen, Som av Hr. Stiftbefalnings-Mand von Nissen var be-
fallet at føre Protocollen her, hørte man ej at kunde hidckomme, formedelst af nogle dagers
Reignveæir Dragaas jisen i Guldahlen Skal være opgaaet, og til Lands ej er at fremckomme.
[node 6098, side 1]

Alternativ 1: <Uspesifisert skriver>
Alternativ 2: <von Nissen>
Alternativ 3: <Skal ikke analyseres>
Alternativ 4: <Peter Schnitler>
Alternativ 5: <Romling>
Alternativ 6: <Schnitler>
Alternativ 7: <Schnitler>

Skriv nummer for forteller, blank for å søke (x avslutter innlegging): 1

Hvorfor det?

Til biesiddere i denne Rætt Mødte som efter directeur Borgrevings foranstaltning vare
tilsagde og tilforn havde Siddet Som laugRættis Mænd i Rætten, 2de Mænd, boendes her
paa Røraas Platz Nafnl: Joen Andersen 54 aar gammel og Elling Svendsen bælgmager 60
aar gammel.
[node 6118, side 1]

Alternativ 1: <Uspesifisert skriver>
Alternativ 2: <Borgrevings>
Alternativ 3: <Joen Andersen>
Alternativ 4: <Elling Svendsen>
Alternativ 5: <Skal ikke analyseres>
Alternativ 6: <von Nissen>
Alternativ 7: <Peter Schnitler>
Alternativ 8: <Romling>
Alternativ 9: <Schnitler>

Skriv nummer for forteller, blank for å søke (x avslutter innlegging): 1

Hvorfor det?

Af Viidner, som skulde være de Kyndigste paa grændserne, Fandtes og fremstillede
Sig her for Rætten 1: Ole Larsen Riise, 2: Peder Andersen Dahlen og 3: Anders Peder-
sen Dahl.
[node 6140, side 1]

Alternativ 1: <Uspesifisert skriver>
Alternativ 2: <Ole Larsen Riise>
Alternativ 3: <Peder Andersen Dahlen>
Alternativ 4: <Anders Pedersen Dahl>
Alternativ 5: <Skal ikke analyseres>
Alternativ 6: <Borgrevings>
Alternativ 7: <Joen Andersen>
Alternativ 8: <Elling Svendsen>
Alternativ 9: <von Nissen>

Skriv nummer for forteller, blank for å søke (x avslutter innlegging): 1
```

```
Hvorfor det?

For dennem blef høyst bemte Kongl: Ordre lydelig oplæst og dernest viidnerne for-
manet, at udsiige, hvad som Rætt og Riigtigt var, og dennem beckiendt Om grændze Gan-
gen og Fieldene imellem dette Riige Norge og det Riige Sværig paa denne Kant; hvor paa
dem og Eedens Forklaring af Loven blef forelæst, og de aflagde deris Corporlig Eed &mdash;
Da og directeuren paa Værcket Leonhardt Borgreving indfandt sig for Rætten 4: viidne.
[node 6154, side 1]

Alternativ 1: <Uspesifisert skriver>
Alternativ 2: <Leonhardt Borgreving>
Alternativ 3: <Skal ikke analyseres>
Alternativ 4: <Ole Larsen Riise>
Alternativ 5: <Peder Andersen Dahlen>
Alternativ 6: <Anders Pedersen Dahl>
Alternativ 7: <Borgrevings>
Alternativ 8: <Joen Andersen>
Alternativ 9: <Elling Svendsen>

Skriv nummer for forteller, blank for å søke (x avslutter innlegging): 1

Hvorfor det?

---> Spørsmaal: 1: Hvad hans nafn er? Resp: Ole Larsen Riise.
[Kort avsnitt; node 6182, side 1]
-----

2: hvor hand er Føed og Af hvad Forældre? Resp: hand er Føed her paa Rør-
aas Platz og var hands Fader en bergsmand her paa værcket.
[node 6190, side 1]

Alternativ 1: <Uspesifisert skriver>
Alternativ 2: <Ole Larsen Riise>
Alternativ 3: <Skal ikke analyseres>
Alternativ 4: <Leonhardt Borgreving>
Alternativ 5: <Ole Larsen Riise>
Alternativ 6: <Peder Andersen Dahlen>
Alternativ 7: <Anders Pedersen Dahl>
Alternativ 8: <Borgrevings>
Alternativ 9: <Joen Andersen>

Skriv nummer for forteller, blank for å søke (x avslutter innlegging): 3

Hvorfor det? Blandet avsnitt

3: Hvor gammel hand er? om er Gift? hvor hand boer eller tilholder ? hvad hands
Næring eller Handteering? Resp: Hand er 71 aar gammel; er gift; hand boer her paa
Platzen; og er een bergsmand ved Værcket her.
[node 6204, side 2]

Alternativ 1: <Skal ikke analyseres>
Alternativ 2: <Uspesifisert skriver>
Alternativ 3: <Ole Larsen Riise>
Alternativ 4: <Leonhardt Borgreving>
Alternativ 5: <Ole Larsen Riise>
Alternativ 6: <Peder Andersen Dahlen>
Alternativ 7: <Anders Pedersen Dahl>
Alternativ 8: <Borgrevings>
Alternativ 9: <Joen Andersen>
```

```
Skriv nummer for forteller, blank for å søke (x avslutter innlegging): 1

Hvorfor det? Blandet avsnitt

4: om hand er beckiendt paa Grænserne her i mellem Guldahls Fogderie i Trond-
hiems Stift og Herjedalen Paa dend Anden Svenske Siide, og Viidere i Søer? Resp: noget
lidt kand hand være beckiendt.
[node 6210, side 2]

Alternativ 1: <Skal ikke analyseres>
Alternativ 2: <Uspesifisert skriver>
Alternativ 3: <Ole Larsen Riise>
Alternativ 4: <Leonhardt Borgreving>
Alternativ 5: <Ole Larsen Riise>
Alternativ 6: <Peder Andersen Dahlen>
Alternativ 7: <Anders Pedersen Dahl>
Alternativ 8: <Borgrevings>
Alternativ 9: <Joen Andersen>

Skriv nummer for forteller, blank for å søke (x avslutter innlegging): 1

Hvorfor det? Blandet avsnitt

5: hvor langt efter Norske Nye Maalte Miile, og i hvad Stræckning denne Røraas
Platz ligger fra Grændsen, og i hvad Fogderie? Resp: Røraas Platz ligger fra høyeste
Rutt Fieldet, Som holdes for et Grændze field, 4re Nye Maalte Norske Miile, og er bemte
Rutten fra Røraas Platz i Nord oest beliggende i Guldahls Fogderie.
[node 6228, side 2]

Alternativ 1: <Skal ikke analyseres>
Alternativ 2: <Uspesifisert skriver>
Alternativ 3: <Ole Larsen Riise>
Alternativ 4: <Leonhardt Borgreving>
Alternativ 5: <Ole Larsen Riise>
Alternativ 6: <Peder Andersen Dahlen>
Alternativ 7: <Anders Pedersen Dahl>
Alternativ 8: <Borgrevings>
Alternativ 9: <Joen Andersen>

Skriv nummer for forteller, blank for å søke (x avslutter innlegging): 1

Hvorfor det? Blandet avsnitt
```

I neste innlegging ønsker jeg å legge inn en taler som ikke finnes i lista, så den må søkes opp.

```
6: hvad landets beskaffenhed er imellem Røraas Platz og Grænse Fieldet, at forstaa;
om der er Skoug, vande, Elfve, Fielde, Myhr el: Morast, dyrcket og bebygget, eller øede
u-frugtbar Land?
[node 6269, side 2]

Alternativ 1: <Skal ikke analyseres>
Alternativ 2: <Uspesifisert skriver>
Alternativ 3: <Ole Larsen Riise>
Alternativ 4: <Leonhardt Borgreving>
Alternativ 5: <Ole Larsen Riise>
Alternativ 6: <Peder Andersen Dahlen>
Alternativ 7: <Anders Pedersen Dahl>
Alternativ 8: <Borgrevings>
```

```
Alternativ 9: <Joen Andersen>

Skriv nummer for forteller, blank for å søke (x avslutter innlegging):

Skriv inn ett eller flere navn du vil søke på: Forhører

Alternativ 1: <Forhører>

Hvilket navn? (blank for nytt søk) 1

Hvorfor det?

Resp: Det bestaar mest af Myhrer og Morast en hoben vande og Søer, og adskillige
Smaa Fielde og Vohler; dog ligger der hidts og her nogle Smaa bønder-gaarder og Platzer,
Som have deris Næring af Værckets brug, og her er saa godt, Som ingen Skaug i
denne Eign.
[node 6278, side 2]

Alternativ 1: <Forhører>
Alternativ 2: <Skal ikke analyseres>
Alternativ 3: <Uspesifisert skriver>
Alternativ 4: <Ole Larsen Riise>
Alternativ 5: <Leonhardt Borgreving>
Alternativ 6: <Ole Larsen Riise>
Alternativ 7: <Peder Andersen Dahlen>
Alternativ 8: <Anders Pedersen Dahl>
Alternativ 9: <Borgrevings>

Skriv nummer for forteller, blank for å søke (x avslutter innlegging): 4

Hvorfor det?

---> 7: Hvilcke ere de nærmeste Gaarder ved Grændse Fieldene?
[Kort avsnitt; node 6286, side 2]
-----

Resp: Synden i fra at reigne, Saa har et par husmænd for nogle Aar Satt Sig need
ved Elgaaen, som kommer Synden fra det Field Elgaa-Hogna østen for Femund-Søen,
(et par husmænd), Men om de ere der endnu, veed hand icke; 3 Miile fra denne Elgaaen
i Nord boer nærmist en bonde paa en liiden Skatte platz ved den Nordere Femunds viig
paa den østere Siide, væsten for det Field viggelskaftets Syndere Ende, ved Nafn Lasse
Jensen Femund, Som holder een hæst og een oxe. &mdash; Halfanden Miil i Nord fra denne
Lasse Femund ligger een Gammel Hytte-Platz ved dend Nordre Ende af Ferragen-Søe, 1
Miil omtrent i Væster fra høyeste Wigel-Field, bestaaende af en 8te opsiddere, Som have
Smaa Platzer, og holde Gemeenligen hver een el: Toe øxene at Kiøre om Vinteren med.
Herfra Ferragens Platz 1 Miil i Nord ere de nærmeste Bræche gaardene, liggende i Væster
nær under Rutt-Fieldet bestaaende omtrent af 13ten Smaa Gaards parter, hvilcke holde hver
omtrent een hæst el: Kiør-øxene. &mdash; Disse alle opReignede Mænd fra Elgaaen til Bræchen
inclusive Sorterer under Røraas Kiercke-Sogn. &mdash;
[node 6289, side 2]

Alternativ 1: <Ole Larsen Riise>
Alternativ 2: <Forhører>
Alternativ 3: <Lasse Jensen Femund>
Alternativ 4: <Lasse Femund>
Alternativ 5: <Uspesifisert skriver>
Alternativ 6: <Skal ikke analyseres>
Alternativ 7: <Ole Larsen Riise>
Alternativ 8: <Leonhardt Borgreving>
Alternativ 9: <Ole Larsen Riise>
```

```
Skriv nummer for forteller, blank for å søke (x avslutter innlegging): 1

Hvorfor det?

8: Hvilcket er det første Field i Syder, viidnet Kiender, Som ligger østligst el: mest
østlig need til Herjedallen ad den østre Svenske Siide?
[node 6372, side 2]

Alternativ 1: <Ole Larsen Riise>
Alternativ 2: <Uspesifisert skriver>
Alternativ 3: <Skal ikke analyseres>
Alternativ 4: <Forhører>
Alternativ 5: <Lasse Jensen Femund>
Alternativ 6: <Lasse Femund>
Alternativ 7: <Ole Larsen Riise>
Alternativ 8: <Leonhardt Borgreving>
Alternativ 9: <Ole Larsen Riise>

Skriv nummer for forteller, blank for å søke (x avslutter innlegging): 4

Hvorfor det?

Resp: Det første i Syder, som hand Kiender mest østlig at være, er Lang-Fieldet,
Som hører Eire bøygd til. Hvad Fielde Siden i Nord eller Nordvæst ligge fra dette Lang-
Field, der veed hand ingen Særdehlis Nafn paa, førend Man Kommer en 4: Miile i Nord
til Siebru-Field og Brat Riie-Field. Dog er her af Brat Rie-Fieldet det østerste, som dahler
ned til Herjedalens Dahl, og er ingen Field yderligere i øster der fra uden nogle Smaa
Wohler (Som er nogle Smaa berg høyder) ved Foeden af Brat Rie-Fieldet. &mdash;
[node 6379, side 2]

Alternativ 1: <Forhører>
Alternativ 2: <Ole Larsen Riise>
Alternativ 3: <Uspesifisert skriver>
Alternativ 4: <Skal ikke analyseres>
Alternativ 5: <Lasse Jensen Femund>
Alternativ 6: <Lasse Femund>
Alternativ 7: <Ole Larsen Riise>
Alternativ 8: <Leonhardt Borgreving>
Alternativ 9: <Ole Larsen Riise>

Skriv nummer for forteller, blank for å søke (x avslutter innlegging): 2

Hvorfor det?

[...]

---> 2det Viidne: Peder Andersen Dahlen.
[Kort avsnitt; node 6974, side 5]
-----

---> Spørsmaal 1: hvad hands Nafn?
[Kort avsnitt; node 6984, side 5]
-----

---> Resp: Peder Andersen Dahlen.
[Kort avsnitt; node 6987, side 5]
-----
```

```
Til det 2: Resp: er Føed paa dend gaard Dahlen i Røraas Sogn og Guldahls Fogde-
rie; og var hans Fader bonde paa Samme Gaard.
[node 6995, side 5]

Alternativ 1: <Ole Larsen Riise>
Alternativ 2: <Peder Andersen Dahlen>
Alternativ 3: <Uspesifisert skriver>
Alternativ 4: <Skal ikke analyseres>
Alternativ 5: <Peder Andersen Dahlen>
Alternativ 6: <Forhører>
Alternativ 7: <Ole Larsen Riise>
Alternativ 8: <Lasse Femunden>
Alternativ 9: <Wibe>

Skriv nummer for forteller, blank for å søke (x avslutter innlegging): 2

Hvorfor det?
```

I neste innlegging velges feil navn på taleren, senere vil det vises hvordan dette kan korrigeres. Navnet som velges er ikke koblet til registeret, så registerinnførselen må søkes opp. I søket framkommer to ikke-reelle treff i tillegg til det rette navnet. Dette skyldes svakheter i implementasjonen av det soundex-baserte søket.

```
Til det 3die: Resp: er 59 aar gammel; er gift og har 8te levendis børn; hand boer
og er bonde paa Forbemte Gaard Dahl; Ernærer Sig med Kuld og bergsveeds leverance til
Røraas værck.
[node 7010, side 5]

Alternativ 1: <Peder Anderssen Dahlen>
Alternativ 2: <Ole Larsen Riise>
Alternativ 3: <Uspesifisert skriver>
Alternativ 4: <Skal ikke analyseres>
Alternativ 5: <Peder Andersen Dahlen>
Alternativ 6: <Peder Andersen Dahlen>
Alternativ 7: <Forhører>
Alternativ 8: <Ole Larsen Riise>
Alternativ 9: <Lasse Femunden>

Skriv nummer for forteller, blank for å søke (x avslutter innlegging): 9

Mangler registerobjekt til pnavn-node: 6886. Vi prøver å lage kobling...

Skriv inn ett eller flere navn du vil søke på: Femunden

Alternativ 1: <Wind, N. O.>
Alternativ 2: <N. O. Wind>
Alternativ 3: <Lasse Jensen Femund>

Hvilket navn? (blank for nytt søk) 3

Hvorfor det?

---> Til det 4: Resp: Hand er beckiendt noget liidet til SvuKue-Field, viidere iche be-
---> ckiendt i Syder;
[Kort avsnitt; node 7036, side 5]
-----

---> Viidnet tilspørges da hvor Stoert SvuKue Fieldet er Fra Syder i Nord og fra øst
```

```
---> i væst?
[Kort avsnitt; node 7047, side 5]
-----

Resp: hand har vel værit paa den eene væstre Siide af SvuKue Fieldet, men iche
Faret der over, hvor fore hand iche egentlig Kand Siige, hvor Stoert det er. Men det veed
hand at SvuKue Field, dets Foed ligger i øster omtrent 1/4 Miil-veigs fra Femund-Søen.
[node 7055, side 5]

Alternativ 1: <Lasse Jensen Femund>
Alternativ 2: <Peder Anderssen Dahlen>
Alternativ 3: <Uspesifisert skriver>
Alternativ 4: <Skal ikke analyseres>
Alternativ 5: <Ole Larsen Riise>
Alternativ 6: <Peder Andersen Dahlen>
Alternativ 7: <Peder Andersen Dahlen>
Alternativ 8: <Forhører>
Alternativ 9: <Ole Larsen Riise>

Skriv nummer for forteller, blank for å søke (x avslutter innlegging): x

Lagt inn taler på noen avsnitt.
```

B.4.2 Endre taler på et avsnitt

Her endres taler på et avsnitt, ny taler søkes opp. Også her kommer noen irrelevante treff opp på grunn av svakheter i implementasjonen av det soundex-baserte søkesystemet.

```
Valg: 3

Endre taler på avsnitt med node nummer: 7010

Til det 3die: Resp: er 59 aar gammel; er gift og har 8te levendis børn; hand boer
og er bonde paa Forbemte Gaard Dahl; Ernærer Sig med Kuld og bergsveeds leverance til
Røraas værck.
[node 7010, side 5]
Taler: Lasse Jensen Femund. Kategori: bonde.
-----

Endre taler til dette avsnittet.

Skriv inn ett eller flere navn du vil søke på: dahlen

Alternativ 1: <Wind, N. O.>
Alternativ 2: <Peder Olsen d. e.>
Alternativ 3: <Peder Anderssen Dahlen>
Alternativ 4: <N. O. Wind>
Alternativ 5: <Anders Pedersen Dahl>
Alternativ 6: <Anders Ammondsen d. y.>
Alternativ 7: <Anders Ammondsen d. e.>

Hvilket navn? (blank for nytt søk) 3

Hvorfor det? Feil navn var angitt
```

B.4.3 Vis avsnittene med taler

Valg: 2

[...]

WIDNERS EXAMINATION
OVER
GRENDSERNE IMELLEM NORGE NORDENFIELDS OG SVERRIGE
ANNO — 1742
PAA VINTERFØRET 1 VOLUMEN.
[node 5965, side 1]
Taler: Uspesifisert skriver. Kategori: embedsmann.

GRÆNDSE COMMISSIONS PROTOKOLL
Over De Tagne Viidner
Begyndt Paa Røraas Dend 16de Apriilis 1742ve Continueret d: 17de ibdm. paa
Koyen ved Ferragen dend 18 og 19de ibm, i Bræchen d: 21de Apriil:
I GULDALS Fogderie.
[node 5983, side 1]
Taler: Uspesifisert skriver. Kategori: embedsmann.

Efter deris Kongl: Maysts allernaadigste befalning til Major Peter Schnitler av 16
Martj Sistleeden og der paa Grundede Jnstrux fra Hr. Oberste og Kongelige grændze
Comissaire Romling til bemelte Major Schnitler af 31: ejusdem, hvilcke begge Documenter
bielegges under Lit: a: og B: har Major Schnitler begivet sig paa Reisen fra Trondhiem til
Grændzerne d: 7 Apriil nest efter og efter foregangne Reqvisition til vedckommende øfrig-
hed og betiendtere beckyndt paa Røraas først at sætte Rætten dend 16 Apriilis Nest efter.
[node 6024, side 1]
Taler: Uspesifisert skriver. Kategori: embedsmann.

Soerenskriiveren i Guldahlen, Som av Hr. Stiftbefalnings-Mand von Nissen var be-
fallet at føre Protocollen her, hørte man ej at kunde hidckomme, formedelst af nogle dagers
Reignveæir Dragaas jisen i Guldahlen Skal være opgaaet, og til Lands ej er at fremckomme.
[node 6098, side 1]
Taler: Uspesifisert skriver. Kategori: embedsmann.

Til biesiddere i denne Rætt Mødte som efter directeur Borgrevings foranstaltning vare
tilsagde og tilforn havde Siddet Som laugRættis Mænd i Rætten, 2de Mænd, boendes her
paa Røraas Platz Nafnl: Joen Andersen 54 aar gammel og Elling Svendsen bælgmager 60
aar gammel.
[node 6118, side 1]
Taler: Uspesifisert skriver. Kategori: embedsmann.

Af Viidner, som skulde være de Kyndigste paa grændserne, Fandtes og fremstillede
Sig her for Rætten 1: Ole Larsen Riise, 2: Peder Andersen Dahlen og 3: Anders Peder-
sen Dahl.
[node 6140, side 1]
Taler: Uspesifisert skriver. Kategori: embedsmann.

For dennem blef høyst bemte Kongl: Ordre lydelig oplæst og dernest viidnerne for-
manet, at udsiige, hvad som Rætt og Riigtigt var, og dennem beckiendt Om grændze Gan-

```
gen og Fieldene imellem dette Riige Norge og det Riige Sværig paa denne Kant; hvor paa
dem og Eedens Forklaring af Loven blef forelæst, og de aflagde deris Corporlig Eed &mdash;
Da og directeuren paa Værcket Leonhardt Borgreving indfandt sig for Rætten 4: viidne.
[node 6154, side 1]
Taler: Uspesifisert skriver. Kategori: embedsmann.
-----

Spørsmaal: 1: Hvad hans nafn er? Resp: Ole Larsen Riise.
[node 6182, side 1]

2: hvor hand er Føed og Af hvad Forældre? Resp: hand er Føed her paa Rør-
aas Platz og var hands Fader en bergsmand her paa værcket.
[node 6190, side 1]
Taler: Skal ikke analyseres. Kategori: .
-----

3: Hvor gammel hand er? om er Gift? hvor hand boer eller tilholder ? hvad hands
Næring eller Handteering? Resp: Hand er 71 aar gammel; er gift; hand boer her paa
Platzen; og er een bergsmand ved Værcket her.
[node 6204, side 2]
Taler: Skal ikke analyseres. Kategori: .
-----

[...]
```

B.4.4 Vis statistikk for alle talere

```
Valg: 4

Navn 342: Peder Anderssen Dahlen (bonde)
Antall avsnitt: 1. Lengde: 125 tegn. Snittlengde: 125 tegn. Tegn pr. snavn: 125.
-----

Navn 313: Ole Larsen Riise (arbeider)
Antall avsnitt: 13. Lengde: 6704 tegn. Snittlengde: 516 tegn. Tegn pr. snavn: 516.
-----

Navn 241: Lasse Jensen Femund (bonde)
Antall avsnitt: 1. Lengde: 181 tegn. Snittlengde: 181 tegn. Tegn pr. snavn: 181.
-----

Navn 3: Skal ikke analyseres
Antall avsnitt: 4. Lengde: 853 tegn. Snittlengde: 213 tegn. Tegn pr. snavn: 213.
-----

Navn 2: Forhører (embedsmann)
Antall avsnitt: 7. Lengde: 1008 tegn. Snittlengde: 144 tegn. Tegn pr. snavn: 144.
-----

Navn 1: Uspesifisert skriver (embedsmann)
Antall avsnitt: 7. Lengde: 1988 tegn. Snittlengde: 284 tegn. Tegn pr. snavn: 284.
-----

Kategori arbeider: Antall avsnitt: 13. Lengde: 6704 tegn. Snittlengde: 515 tegn.
Kategori bonde: Antall avsnitt: 2. Lengde: 306 tegn. Snittlengde: 153 tegn.
Kategori : Antall avsnitt: 4. Lengde: 853 tegn. Snittlengde: 213 tegn.
Kategori embedsmann: Antall avsnitt: 14. Lengde: 2996 tegn. Snittlengde: 214 tegn.
```

B.5 Frekvensanalyse

```
Valg: d

DELMENY: Frekvensanalyse

1. Lag leksikon for alle avsnitt tilordnet taler.
2. Lag samlet frekvenstabell for alle avsnitt tilordnet taler.
3. Sammenlign en person med snittet. Resultat sortert etter avstand.
4. Sammenlign en kategori med snittet. Resultat sortert etter avstand.
5. Sammenlign en person med snittet. Resultat sortert etter total frekvens.
6. Sammenlign en kategori med snittet. Resultat sortert etter total frekvens.
7. Sammenlign alle kategoriers frekvens med snittet.
   Resultat sortert etter total frekvens.
8. Vis alle kategoriers frekvens. Resultat sortert etter total frekvens.
9. Vis alle kategoriers sum. Resultat sortert etter total frekvens.
10. Vis alle personers frekvens. Resultat sortert etter total frekvens.
11. Vis alle personers ordlengdesnitt.
12. Vis alle personers ordlengdesnitt med signatur.
13. Skriv alle personers frekvens i lisp-statistikk-format.
   Resultat sortert etter total frekvens. Uaktuelle kategorier utelates.
14. Skriv alle personers frekvens i lisp-statistikk-format med lagring til fil.
   Resultat sortert etter total frekvens. Uaktuelle kategorier utelates.
15. Skriv alle personers frekvens i lisp-statistikk-format.
   Resultat sortert etter total frekvens. Uaktuelle kategorier utelates.
   LoiczView-format.
16. Skriv alle personers frekvens i SPSS-format.
   Resultat sortert etter total frekvens. Uaktuelle kategorier utelates.
Q. Ut
```

Før analyser kan gjøres, må menyvalg 1 og 2 kjøres. Her vises korte utdrag av kjøringsresultatene for de andre menyvalgene.

B.5.1 Sammenlign en person med snittet, sortert etter avstand

```
Valg: 3

Skriv inn pnavnnummer eller blank for søk på navnet: 448

Sammenligning av person 448 (Zacharias Olsen) med snittet:

Overrepresentasjon i argument 1:
Avstand 0.02564: at
Avstand 0.02391: hand
Avstand 0.01913: det
Avstand 0.01257: finner
Avstand 0.01042: have
Avstand 0.01023: ham
Avstand 0.01017: som
```

```
Avstand 0.01010: vil
Avstand 0.01001: spørsmaal
Avstand 0.00999: hands

Underrepresentasjon i argument 1:
Avstand -0.01046: fra
Avstand -0.00969: i
Avstand -0.00331: er
Avstand -0.00268: ved
Avstand -0.00243: der
Avstand -0.00145: dette
Avstand -0.00115: pnavn
Avstand -0.00114: de
Avstand -0.00108: hvor
Avstand -0.00083: norske

Antall ord i argument 1: 281; argument 2: 166841
--------------------------
```

B.5.2 Sammenlign en kategori med snittet, sortert etter avstand

```
Valg: 4

Disse kategoriene finnes:

Alternativ 1: <adel>
Alternativ 2: <arbeider>
Alternativ 3: <bonde>
Alternativ 4: <embedsmann>
Alternativ 5: <fastboende>
Alternativ 6: <funksjonær>
Alternativ 7: <håndverker>
Alternativ 8: <jeger>
Alternativ 9: <kirke>
Alternativ 10: <kvæn>
Alternativ 11: <lagrett>
Alternativ 12: <offiser>
Alternativ 13: <politi>
Alternativ 14: <reindriftssame>
Alternativ 15: <sjøsame>
Alternativ 16: <skole>
Alternativ 17: <soldat>

Hvilken kategori? 3

[...]

Sammenligning av personer i kategori bonde med snittet:
```

```
Overrepresentasjon i argument 1:
Avstand 0.00935: dend
Avstand 0.00924: hand
Avstand 0.00475: der
Avstand 0.00463: siide
Avstand 0.00397: det
Avstand 0.00393: denne
Avstand 0.00393: ligger
Avstand 0.00356: een
Avstand 0.00350: væster
Avstand 0.00347: nye

Underrepresentasjon i argument 1:
Avstand -0.00608: den
Avstand -0.00365: til
Avstand -0.00307: pnavn
Avstand -0.00299: side
Avstand -0.00279: norske
Avstand -0.00242: de
Avstand -0.00208: vester
Avstand -0.00206: næss
Avstand -0.00202: med
Avstand -0.00190: i

Antall ord i argument 1: 30645; argument 2: 166841
--------------------------
```

B.5.3 Sammenlign en person med snittet, sortert etter total frekvens

Her søkes først en person opp, men den personen har ikke tilordnet noen avsnitt. Rutinen kjøres på nytt, da med en person valgt på nummer. Den siste gangen blir det resultater.

```
Valg: 5

Skriv inn pnavnnummer eller blank for søk på navnet:

Skriv inn ett eller flere navn du vil søke på: Sjur

Alternativ 1: <Sjur (Siur) Sevaldsen>
Alternativ 2: <Sjur (Siur) Pedersen>
Alternativ 3: <Sjur (Siur) Mathisen (Mathiasen)>
Alternativ 4: <Jens Andersen Skreifjord>
Alternativ 5: <Uspesifisert skriver *>

Hvilket navn? (blank for nytt søk) 3

Navnet 407 har ikke tilknyttet avsnitt.
```

```
Valg: 5

Skriv inn pnavnnummer eller blank for søk på navnet: 448

Sammenligning av person 448 (Zacharias Olsen) med snittet:

Overrepresentasjon i argument 1:
Avstand 0.00220: snavn
Avstand 0.00068: og
Avstand -0.00969: i
Avstand 0.00889: til
Avstand -0.01046: fra
Avstand -0.00331: er
Avstand 0.00769: af
Avstand 0.00246: paa
Avstand 0.02564: at
Avstand -0.01299: miil
Avstand 0.01913: det
Avstand -0.00114: de
Avstand 0.01017: som
Avstand -0.00961: den
Avstand 0.00113: for
Avstand -0.00871: med
Avstand -0.00752: en
Avstand -0.00628: 1
Avstand -0.00268: ved

Antall ord i argument 1: 281; argument 2: 166841
--------------------------
```

B.5.4 Sammenlign en kategori med snittet, sortert etter total frekvens

```
Valg: 6

Disse kategoriene finnes:

Alternativ 1: <adel>
Alternativ 2: <arbeider>
Alternativ 3: <bonde>
Alternativ 4: <embedsmann>
Alternativ 5: <fastboende>
Alternativ 6: <funksjonær>
Alternativ 7: <håndverker>
Alternativ 8: <jeger>
Alternativ 9: <kirke>
Alternativ 10: <kvæn>
Alternativ 11: <lagrett>
Alternativ 12: <offiser>
Alternativ 13: <politi>
```

```
Alternativ 14: <reindriftssame>
Alternativ 15: <sjøsame>
Alternativ 16: <skole>
Alternativ 17: <soldat>


Hvilken kategori? 3


Sammenligning av personer i kategori bonde med snittet:

Overrepresentasjon i argument 1:
Avstand 0.00209: snavn
Avstand 0.00014: og
Avstand -0.00190: i
Avstand -0.00365: til
Avstand 0.00327: fra
Avstand 0.00347: er
Avstand 0.00047: af
Avstand -0.00189: paa
Avstand 0.00091: at
Avstand 0.00120: miil
Avstand 0.00397: det
Avstand -0.00242: de
Avstand 0.00338: som
Avstand -0.00608: den
Avstand 0.00008: for
Avstand -0.00202: med
Avstand -0.00024: en
Avstand -0.00057: 1
Avstand 0.00006: ved

Antall ord i argument 1: 30645; argument 2: 166841
--------------------------
```

B.5.5 Sammenlign alle kategoriers frekvens med snittet, sortert etter total frekvens

```
Valg: 7


                          soldat      skole    sjøsame  reindrift [...]
               166841       3305        168      22372      15640 [...]
snavn           13883 -0.0051474 -0.0355919 +0.0066335 +0.0063670 [...]
og               6418 +0.0035897 -0.0146582 -0.0028876 -0.0006161 [...]
i                6366 -0.0094118 +0.0094630 +0.0007318 -0.0021586 [...]
til              3266 +0.0034199 -0.0076708 -0.0006680 +0.0010766 [...]
fra              2933 -0.0045690 -0.0116272 +0.0020878 -0.0017228 [...]

[...]
```

B.5.6 Vis alle kategoriers frekvens, sortert etter total frekvens

```
Valg: 8

                      soldat      skole    sjøsame  reindrift [...]
              166841      3305        168      22372      15640 [...]
snavn          13883 +0.0780635 +0.0476190 +0.0898445 +0.0895780 [...]
og              6418 +0.0420575 +0.0238095 +0.0355802 +0.0378517 [...]
i               6366 +0.0287443 +0.0476190 +0.0388879 +0.0359974 [...]
til             3266 +0.0229955 +0.0119048 +0.0189076 +0.0206522 [...]
fra             2933 +0.0130106 +0.0059524 +0.0196674 +0.0158568 [...]

[...]
```

B.5.7 Vis alle kategoriers sum, sortert etter total frekvens

```
Valg: 9

                      soldat      skole    sjøsame  reindrift [...]
              166841      3305        168      22372      15640 [...]
snavn          13883       258          8       2010       1401 [...]
og              6418       139          4        796        592 [...]
i               6366        95          8        870        563 [...]
til             3266        76          2        423        323 [...]
fra             2933        43          1        440        248 [...]

[...]
```

B.5.8 Vis alle personers frekvens, sortert etter total frekvens

Personnummer i første rekke, totale frekvenser i i annen rekke, deretter enkeltord.

```
Valg: 10

                          1          2          3         13 [...]
              166841      68531      2785       5196       201 [...]
snavn          13883 +0.0830427 +0.0380610 +0.0706313 +0.1044776 [...]
og              6418 +0.0379536 +0.0488330 +0.0311778 +0.0447761 [...]
i               6366 +0.0412660 +0.0193896 +0.0327175 +0.0149254 [...]
til             3266 +0.0201661 +0.0175943 +0.0142417 +0.0348259 [...]
fra             2933 +0.0170288 +0.0157989 +0.0065435 +0.0049751 [...]

[...]
```

B.5.9 Vis alle persones ordlengdesnitt

```
Valg: 11
```

```
Ordlengede for alle navn:

    1 Uspesifisert skriver           embedsmann      +4.5094776
    2 Forhører                       embedsmann      +4.6696587
    3 Skal ikke analyseres                           +3.6805234
   13 Ammond Olsen                   reindriftssame  +4.4676620
   14 Anders Ammondsen d. e.         reindriftssame  +4.2639594
   16 Anders Aslaksen                reindriftssame  +4.0677967
   21 Anders Henningsen Miøsødal     bonde           +4.3823530
   23 Anders Iversen Borgoesen       bonde           +4.2086167
   30 Anders Olsen                   sjøsame         +4.0733814
   34 Anders Pedersen Dahl           bonde           +4.1864243
   35 Anders Poulsen                 sjøsame         +4.4926630
   39 Andfind Ingebrigtsen Gran      bonde           +4.3200000
   51 Baldtzer Nielsen               NIL             +4.6987953
   54 Bendt (Bent) Hansen Øsnaar     bonde           +4.3660820
   55 Bendt (Bent) Simensen Løføyen  bonde           +4.3701620
   56 Bendt (Bent) Thøresen (Tørre   bonde           +4.1818180
   60 Borgreving, -nk, Leonhardt     funksjonær      +4.3717947
   75 Clement Nielsen                sjøsame         +4.2607260
   77 Clement Samuelsen              kvæn            +4.3260870
   81 Ejnar Skieldbreed              bonde           +4.8154610
   86 Erik (Erich) Bonjækas          sjøsame         +4.5482287
   88 Erik (Erich) Clementsen (-sø   sjøsame         +4.4611110
   90 Erik (Erich) Halvorsen Aas     bonde           +4.1902150
   92 Erik (Erich) Jensen Giedsaas   bonde           +4.3923707
   95 Erik (Erich) Olsen Riisvold    bonde           +4.3086660
   99 Frederik III                   adel            +5.0759716

[...]
```

B.5.10 Vis alle persones ordlengdesnitt med signatur

Signatur betyr at fordelingen mellom ordlengder (1 bokstav, 2 bokstaver, 3 bokstaver osv.) vises etter samlet frekvens på hver linje.

```
Valg: 12

Ordlengede for alle navn:

 1|Uspesifisert skriver |embedsmann     |+4.5094776|+0.000|+6.800|+13.235 [...]
 2|Forhører             |embedsmann     |+4.6696587|+0.000|+4.309|+18.851 [...]
 3|Skal ikke analyseres |               |+3.6805234|+0.000|+25.808|+14.126 [...]
13|Ammond Olsen         |reindriftssame |+4.4676620|+0.000|+2.488|+19.900 [...]

[...]
```

B.5.11 Skriv alle personers frekvens i lisp-stat-format

```
Valg: 13
```

```
(defun skriv-plots ()
(def kat (list 4 4 14 14 14 [...]))
(def snavn (list 0.0830427 0.0380610 0.1044776 0.1421320 0.0677966 [...]))
(def plottene (plot-points kat snavn :variable-labels '("kategori" "frekvens")))
(send plottene :title "snavn")
(send plottene :add-points kat snavn)
(send plottene :draw-text '"snavn" 200 10 1 1)
(send plottene :variable-label 1 '"frekvens snavn")
(def og (list 0.0379536 0.0488330 0.0447761 0.0558376 0.0338983 [...]))
(def plottene (plot-points kat og :variable-labels '("kategori" "frekvens")))
(send plottene :title "og")
(send plottene :add-points kat og)
(send plottene :draw-text '"og" 200 10 1 1)
(send plottene :variable-label 1 '"frekvens og")

[...]
```

B.5.12 Skriv alle personers frekvens i lisp-stat-format med lagring til fil

```
Valg: 14

(defun skriv-plots ()
(def kat (list 4 4 14 14 14 [...]))
(def snavn (list 0.0830427 0.0380610 0.1044776 0.1421320 0.0677966 [...]))
(def plottene (plot-points kat snavn :variable-labels '("kategori" "frekvens")))
(send plottene :title "snavn")
(send plottene :add-points kat snavn)
(send plottene :draw-text '"snavn" 200 10 1 1)
(send plottene :variable-label 1 '"frekvens snavn")
(send plottene :save-image
      '"/hf/hedvig/muspro-u1/oeide/utv/lisp/hoppg/analyse-data/a14_500snavn.ps")
(send plottene :close)
(def og (list 0.0379536 0.0488330 0.0447761 0.0558376 0.0338983 [...]))
(def plottene (plot-points kat og :variable-labels '("kategori" "frekvens")))
(send plottene :title "og")
(send plottene :add-points kat og)
(send plottene :draw-text '"og" 200 10 1 1)
(send plottene :variable-label 1 '"frekvens og")
(send plottene :save-image
      '"/hf/hedvig/muspro-u1/oeide/utv/lisp/hoppg/analyse-data/a14_500og.ps")
(send plottene :close)

[...]
```

Valg 15 brukes ikke.

B.5.13 Skriv alle personers frekvens i SPSS-format

```
Valg: 16
```

```
§ID§|§navn§|§katnr§|§kat§|§snavn§|§og§|§i§ [...]
1|§Uspesifisert skriver§|4|§embedsmann§|0.0830427|0.0379536|0.0412660 [...]
2|§Forhører§|4|§embedsmann§|0.0380610|0.0488330|0.0193896 [...]
13|§Ammond Olsen§|14|§reindriftssame§|0.1044776|0.0447761|0.0149254 [...]

[...]
```

§ *blir erstattet med " før import til SPSS.*

B.6 Naboanalyse

```
Valg: e

DELMENY: Naboanalyse

1. Lag kontekstabeller for alle avsnitt tilordnet taler.
2. Lag samlet kontekstabell for alle avsnitt tilordnet taler.
3. Vis alle personers etter snavn-frekvens. Resultat sortert etter total
   etter snavn-frekvens. Utelater uaktuelle kategorier.
4. Vis alle personers etter snavn-frekvens.
   Resultat sortert etter total etter snavn-frekvens.
   Utelater uaktuelle kategorier og personer med færre enn 25 kontekster.
5. Skriv alle personers etter snavn-frekvens i lisp-statistikk-format.
   Resultat sortert etter total etter snavn-frekvens.
   Uaktuelle kategorier utelates.
6. Skriv alle personers etter snavn-frekvens i SPSS-format.
   Resultat sortert etter total etter snavn-frekvens.
   Uaktuelle kategorier og personer med færre enn 25 kontekster utelates.
7. Skriv etter-kontekst etter snavn + ett eller flere ord for alle personer
   sortert etter kategori.
8. Skriv mellom-kontekst mellom snavn gruppert på navn,
   sortert etter kategori. Maxlengde 10 ord.
9. Skriv mellom-kontekst mellom snavn gruppert og sortert etter kategori.
   Maxlengde 10 ord.
10. Skriv mellom-kontekst mellom snavn gruppert og sortert etter kategori.
    Valgfri maxlengde.
Q. Ut
```

Før analyser kan gjøres, må menyvalg 1 og 2 kjøres. Her vises korte utdrag av kjøringsresultatene for de andre menyvalgene.

B.6.1 Vis alle personers etter snavn-frekvens, sortert etter total etter snavn-frekvens

Personnummer i første rekke, totale frekvenser i i annen rekke, deretter enkeltord.

```
Valg: 3

                1          2         13         14         16 [...]
    14295       5849        154         22         29          4 [...]
```

```
og   1372 +0.0776201 +0.0776201 +0.1883117 +0.2727273 +0.1379310 [...]
i    1109 +0.0926654 +0.0926654 +0.0064935 +0.0000000 +0.0344828 [...]
er    835 +0.0637716 +0.0637716 +0.0064935 +0.0000000 +0.0000000 [...]
NIL   739 +0.0478714 +0.0478714 +0.3376623 +0.0454545 +0.0344828 [...]

[...]
```

B.6.2 Vis alle personer med mer enn 25 kontekser sin etter snavn-frekvens, sortert etter total etter snavn-frekvens

Personnummer i første rekke, totale frekvenser i i annen rekke, deretter enkeltord.

```
Valg: 4

                  1          2         14         23         30 [...]
    14295       5849        154         29         48         47 [...]
og   1372 +0.0776201 +0.0776201 +0.1883117 +0.1379310 +0.2083333 [...]
i    1109 +0.0926654 +0.0926654 +0.0064935 +0.0344828 +0.0416667 [...]
er    835 +0.0637716 +0.0637716 +0.0064935 +0.0000000 +0.0625000 [...]
NIL   739 +0.0478714 +0.0478714 +0.3376623 +0.0344828 +0.0208333 [...]

[...]
```

B.6.3 Vis alle personers etter snavn-frekvens i lisp-stat-format, sortert etter total etter snavn-frekvens

```
Valg: 5

(defun skriv-plots ()
(def kat (list 4 4 14 14 14 [...]))
(def og (list 0.0776201 0.1883117 0.2727273 0.1379310 0.0000000 [...]))
(def plottene (plot-points kat og :variable-labels '("kategori" "frekvens")))
(send plottene :title "og")
(send plottene :add-points kat og)
(send plottene :draw-text '"og" 200 10 1 1)
(send plottene :variable-label 1 '"frekvens og")
(def i (list 0.0926654 0.0064935 0.0000000 0.0344828 0.0000000 [...]))
(def plottene (plot-points kat i :variable-labels '("kategori" "frekvens")))
(send plottene :title "i")
(send plottene :add-points kat i)
(send plottene :draw-text '"i" 200 10 1 1)
(send plottene :variable-label 1 '"frekvens i")

[...]
```

B.6.4 Vis alle personers etter snavn-frekvens i SPSS-format, sortert etter total etter snavn-frekvens

```
Valg: 6
```

```
§ID§|§navn§|§katnr§|§kat§|§og§|§i§|§er§|§NIL§ [...]
1|§Uspesifisert skriver§|4|§embedsmann§|0.0776201|0.0926654|0.0637716 [...]
2|§Forhører§|4|§embedsmann§|0.1883117|0.0064935|0.0064935|0.3376623 [...]
14|§Anders Ammondsen d. e.§|14|§reindriftssame§|0.1379310|0.0344828 [...]
23|§Anders Iversen Borgoesen§|3|§bonde§|0.2083333|0.0416667|0.0625000 [...]
30|§Anders Olsen§|15|§sjøsame§|0.1276596|0.0851064|0.0212766|0.0638298 [...]

[...]
```

§ *blir erstattet med " før import til SPSS.*

B.6.5 Skriv etter-kontekst etter snavn og ett eller flere ord

```
Valg: 7
Skriv null eller flere ord kontekst etter SNAVN, x for å avslutte > i
> x

Kontekst <SNAVN i>

[...]

Kategori bonde:

436 Tørris Nilsen Suul:
SNAVN i begyndelsen i væster Siiden i Nord-væst 1 1/2 miil (avsn 25281)
SNAVN i Nord mdash (avsn 25991)
SNAVN i SNAVN mdash (avsn 25711)
SNAVN i SNAVN og denne SNAVN og Elfv giiver SNAVN deris (avsn 25407)
SNAVN i SNAVN og SNAVN Da det dog er en beckiendt (avsn 25117)
SNAVN i SNAVN omtrendt 5 1/2 Svenske Miile Saa have de (avsn 25541)
SNAVN i SNAVN Som før er beskreeven i SNAVN ligger i (avsn 25472)
SNAVN i Søer og paa SNAVN af SNAVN i Nord mdash (avsn 25991)
SNAVN i væster der fra og 1/2 Miil i Søer fra (avsn 25355)
SNAVN i øster 3/8 miil her ved denne SNAVN er det (avsn 25117)
SNAVN i øster der fra thi af hans forfædre har hand (avsn 25991)
SNAVN i øster følger ett field kaldet SNAVN dette field Stræcker (avsn 25117)
SNAVN i øster liggendes er ett lande-mærcke imellem SNAVN og SNAVN (avsn 25117)
SNAVN i øster Saaleedes er denne fløtning af Mærcke- Stolpen nogle (avsn 25991)

422 Thomas (Thomes, Tomes) Jacobsen Helgaasen:
SNAVN i dette SNAVN hvad nu det 1te viidne paa disse (avsn 26532)
SNAVN i fieldet Paa disse Skoug- dale i øster kommer ett (avsn 26427)
SNAVN i Nord Siiger han at nærmist i Nord fra SNAVN (avsn 26647)
SNAVN i SNAVN hvor fieldet opstiiger i Væster indtil det field (avsn 26647)
SNAVN i SNAVN ved dette 3die Spørsmaal har udsagt om Land- (avsn 26532)
SNAVN i Sydost ligger Een Skoug-aass kaldet SNAVN med Gran- og (avsn 26427)
SNAVN i øster og i Sæhr indtil Samme bemte SNAVN Norderste (avsn 26647)

[...]
```

```
Kategori reindriftssame:

448 Zacharias Olsen:
SNAVN i øster til SNAVN og SNAVN paa dend væstere Ende (avsn 46530)

427 Thomas (Thomes, Tomes) Tomesen (Brede-Thomes):
SNAVN i Samme Præstegield dend Fin PNAVN Der fra er dend (avsn 47170)
SNAVN i SNAVN Hand har af dette Sit Rom paa SNAVN (avsn 47021)
SNAVN i SNAVN og om Sommern SNAVN i SNAVN Hand har (avsn 47021)
SNAVN i SNAVN Sidder dend Finn PNAVN j SNAVN i Samme (avsn 47170)
SNAVN i Søer til imod Skougen Som hand Vil meene Skal (avsn 47021)
SNAVN i væster til SNAVN J Nord til SNAVN i Søer (avsn 47021)

[...]
```

B.6.6 Skriv mellom-kontekst mellom snavn gruppert på navn, sortert på kategori, makslengde 10 ord

```
Valg: 8

Kontekster mellom snavn, lengde opp til 10 tegn.

[...]

Kategori bonde:

436 Tørris Nilsen Suul: (100 kontekster)
(avsn 25631)
1 1/2 miil hvor fra landevejen viidere gaar need ad (avsn 26063)
1/8 miil østen for (avsn 25541)
3 field Miile herfra beliggende Denne (avsn 25281)
7 miile fra (avsn 26085)
af (avsn 25991)
bestaar af Gran og Furru Skoug Stræckende Sig til (avsn 25814)
Dend Elfv Som af denne Stor-Søe udgaar i øster kaldes (avsn 25730)
Der efter Rinder denne (avsn 25281)
der hvor gran-Skoug tager paa til mod (avsn 25730)
der udkommer af dend Søe (avsn 25541)
det andet lande-Mærcke Viidnet kiender at være imel- lem (avsn 25711)
Dette (avsn 25864)
dog er Flatere og Slettere Dend Nordre Siide af (avsn 25864)
een gaard der fra i Nord 1 miil (avsn 25472)
Eet Støcke fra (avsn 25281)
eller paa (avsn 26085)
en half miil fra (avsn 25117)
er 1 1/2 miil i øster alle liggendes efter (avsn 25541)
er 1 1/2 Miil Saa er dend første nembl (avsn 25541)
er det at de Svenske gierne vil have grændse-Mærcket imellem (avsn 25117)
er det Samme paa østere og væstere Siide som ved (avsn 25915)
er fra disse (avsn 26063)
```

```
er liige som af (avsn 25864)
er Myhret og blødt med nogen Smaa biercke-Skoug indtil forbi (avsn 25730)
er Skouget og Myhret 1/4 miil vejs ind- til (avsn 25864)
gaa liige neer i Nord til (avsn 25665)
gaar Linien beent i Nord Saa at denne (avsn 25864)
har dend Norske øfrighed ladet dend fløtte til (avsn 25991)
har fra Sig i øster mdash 1/8 miil til (avsn 25117)
have dend Søe (avsn 25238)
have undertiden deris tilhold i disse (avsn 26121)
her fra (avsn 25238)
hvilcket de Svenske vil tilEigne Sig indtil (avsn 25991)
hvis leje og landskab paa Siiderne see 1 Vidne i (avsn 25665)
i (avsn 25117)
i (avsn 25407)
i (avsn 25472)
i (avsn 25541)
i (avsn 25711)
i Søer og paa (avsn 25991)
i væster der fra og 1/2 Miil i Søer fra (avsn 25355)
i øster 3/8 miil her ved denne (avsn 25117)
i øster følger ett field kaldet (avsn 25117)
i øster liggendes er ett lande-mærcke imellem (avsn 25117)
ind i (avsn 25281)

[...]
```

B.6.7 Skriv mellom-kontekst mellom snavn gruppert og sortert på kategori, makslengde 10 ord

```
Valg: 9

Kontekster mellom snavn, lengde opp til 10 tegn

[...]

Kategori bonde:
 (1611 kontekster)

(avsn 10327)
(avsn 10398)
(avsn 10398)
(avsn 10770)
(avsn 10770)
(avsn 10985)
[...]
1 1/2 miil breed østen for disse (avsn 17598)
1 1/2 miil hvor fra landevejen viidere gaar need ad (avsn 26063)
1 1/2 Nye Miile adskildt Fra østerste gaard (avsn 28338)
1 liiden field-Miil ved Elven (avsn 29154)
1 liiden Miil ligger dend gaard (avsn 32599)
```

```
1 miil dend Gaard (avsn 42677)
1 Miil fra (avsn 8501)
1 miil fra (avsn 8501)
1 Miil her fra igiennem (avsn 8501)
1 miil hvor dette viidne Selv har værit paa Dette (avsn 10724)
1 Miil i NordNordvest Fra (avsn 127919)
1 Miil i Syd ost ind i (avsn 12033)
1 miil mdash Forbemte (avsn 27903)
1 Miil Norden for disse (avsn 23629)
1 miil og her fra til (avsn 11248)
1 Miil og Siiden paa begge Siider af (avsn 30583)
1 miil omtrent lang og breed Synden for (avsn 17416)
1 Miil Paa denne (avsn 26427)
1 miil vejs omtrent i væster ud i (avsn 12988)
1 Ny Fierding Miil der over er (avsn 33454)
1 Nye Miil breed breed er dette (avsn 30706)
1 Nye Miil i Nordvæst til nærmeste grændse-mærcke (avsn 34108)
1 Søemil i OstSydost er til (avsn 127972)
1/2 miil fra (avsn 32599)
1/2 nye Miil i Søer og fra (avsn 33225)
1/4 miil Norden for (avsn 42993)
1/4 miil vejs i væster ind i (avsn 23471)
1/4 miil vejs i øster Af denne (avsn 24051)
1/4 miil vejs i øster i (avsn 26806)
1/4 Miil væsten for de 3 høye (avsn 12674)
1/4 Nye Miil i Søer i (avsn 32991)
1/4 Nye Miil Norden for det vand (avsn 32991)
1/4 vejs ind i (avsn 31348)
1/4 vejs Som meldt fra (avsn 23731)
1/8 dehl Miil omtrændt breed og Kal- des dend (avsn 9044)
1/8 Miil vejs fra dendne (avsn 27903)
1/8 miil østen for (avsn 25541)
11 Nye Miile Paa dend væstre Siide ad (avsn 34661)
11ve gamle Miile fra Grændse-Mærcket (avsn 42528)
12 field- eller 8 Nye- Miile liggende er dend bøjd (avsn 31699)
13 Smaa (avsn 10360)
2 field-miile i Nord til (avsn 26647)
2 field-Miile i Nord til (avsn 27903)
2 Miile der fra i øster Derfra nu at denne (avsn 23471)
2 miile i væster fra høyeste (avsn 10327)
2 Miile til (avsn 23989)
2 mile i Nord ost fra hin dend Søe (avsn 26738)
2 Nye Miil liggendes i væster fra (avsn 34661)
2 Nye Miile Dog ligger imellem dette (avsn 30706)
2 Nye Miile i væster Denne (avsn 32775)
2de gaar- der vel 2 goede Miile og til (avsn 18148)
3 a 4 Bøsseskud lang rinder i Nord i (avsn 128044)
3 det Field (avsn 16957)
3 field Miile herfra beliggende Denne (avsn 25281)
3 miil derimod fra (avsn 11248)
```

```
3 miile lang i væster og hører (avsn 19322)
3 miile ligger dend gaard (avsn 23629)
4 dett Field (avsn 16957)
4 Nye Miile over der efter dend Skoug dahlen (avsn 30888)
5 gamle eller formeenende 4 Nye Miile ud i (avsn 23471)
6 miile derfra ligge 2de (avsn 16846)
7 miile fra (avsn 26085)
ad (avsn 34485)
af (avsn 23471)
af (avsn 25991)
af (avsn 33945)
af den Søe (avsn 17464)
Af denne (avsn 32599)
af Eett Marcks Skyld der ligger tædt under (avsn 16773)
af Elfven (avsn 19462)
af hvilcken Søe udkommer (avsn 17180)
alleene af Sneaasingerne fiskes Disse 8 (avsn 28338)
Angaaendes landskabet fra Fieldet til (avsn 16773)

[...]
```

B.6.8 Skriv mellom-kontekst mellom snavn gruppert og sortert på kategori, valgfri makslengde

```
Valg: 10

Maksimalt antall ord: 1

Kontekster mellom snavn, lengde opp til 1 tegn

Kategori bonde:
 (404 kontekster)

(avsn 10327)
(avsn 10398)
(avsn 10398)
(avsn 10770)
(avsn 10770)
(avsn 10985)
(avsn 11507)
(avsn 127473)
(avsn 127627)
(avsn 127627)
[...]
ad (avsn 34485)
af (avsn 23471)
af (avsn 25991)
af (avsn 33945)
benytte (avsn 127727)
bruger (avsn 42821)
```

```
dend (avsn 11802)
Dendne (avsn 26806)
denne (avsn 11970)
denne (avsn 27903)
Denne (avsn 27903)
Denne (avsn 28491)
Denne (avsn 28647)
Denne (avsn 32599)
Denne (avsn 32599)
Denne (avsn 32775)
denne (avsn 7718)
denne (avsn 8662)
dernæst (avsn 26957)
Dette (avsn 11878)
Dette (avsn 11970)
dette (avsn 13102)
Dette (avsn 25864)
Dette (avsn 27903)
Dette (avsn 28491)
Dette (avsn 32599)
Dette (avsn 32775)
dette (avsn 34733)
dette (avsn 8328)

[...]
```

B.7 Vis tekst

```
Valg: f

DELMENY: Vis tekst

1. Vis teksten under et subtre.
2. Skriv KWIC-konkordans i HTML-format gruppert etter kategori.
3. Skriv KWIC-konkordans med pnavn og snavn erstattet i HTML-format
   gruppert etter kategori.
4. Skriv KWIC-konkordans med trunkering av ordet med pnavn og snavn
   erstattet i HTML-format gruppert etter kategori.
5. Skriv KWIC-konkordans i HTML-format sortert etter nodenummer.
Q. Ut
```

B.7.1 Vis teksten under et subtre

```
Valg: 1

Nodenummer: 6539

13: Hvilcket Field i Nord følger paa Brat Rie Field som ligger østerst til
Herjedalen?
```

B.7.2 Skriv KWIC-konkordans sortert etter etter-kontekst, gruppert etter kategori

```
Valg: 2

Ordform: bøsseskud

<HTML><HEAD><TITLE>Konkordans bøsseskud</TITLE></HEAD>
<BODY>
<H1>Konkordans bøsseskud</H1>

[...]

<H2>Kategori bonde</H2>

<table>
<tr><td align=right>1/8 miil laang og 2 <td><b>bøsseskud</b><td> breed
    og hænge i nord-væst (avsn 34618)</tr>
<tr><td align=right>hvoraf langfiordelven 3 a 4
    <td><b>bøsseskud</b><td> lang rinder i nord i (avsn 128044)</tr>
<tr><td align=right>1/4 miil lang og 1 <td><b>bøsseskud</b><td> over
    breed (avsn 127128)</tr>
</table>

[...]
```

B.7.3 Skriv KWIC-konkordans med pnavn og snavn erstattet sortert etter etter-kontekst, gruppert etter kategori

Erstatning av pnavn *og* snavn *betyr at alle subtrær under et element av en av typene erstattes med hhv. "PNAVN" og "SNAVN" i konkordansen.*

```
Valg: 3

Ordform: bøsseskud

<HTML><HEAD><TITLE>Konkordans bøsseskud</TITLE></HEAD>
<BODY>
<H1>Konkordans bøsseskud</H1>

[...]

<H2>Kategori bonde</H2>

<table>
<tr><td align=right>1/8 miil laang og 2 <td><b>bøsseskud</b><td> breed
    og hænge i nord-væst (avsn 34618)</tr>
<tr><td align=right>hvoraf snavn 3 a 4 <td><b>bøsseskud</b><td> lang
    rinder i nord i (avsn 128044)</tr>
```

```
<tr><td align=right>1/4 miil lang og 1 <td><b>bøsseskud</b><td> over
    breed (avsn 127128)</tr>
</table>

[...]
```

B.7.4 Skriv KWIC-konkordans med trunkering av ordet med pnavn og snavn erstattet sortert etter etterkontekst, gruppert etter kategori

```
Valg: 4

Ordform: bøsse

<HTML><HEAD><TITLE>Konkordans bøsse</TITLE></HEAD>
<BODY>
<H1>Konkordans bøsse</H1>

[...]

<H2>Kategori bonde</H2>

<table>
<tr><td align=right>miil og er et liidet <td><b>bøsse-skud</b><td>
    breed der af naar elven (avsn 32775)</tr>
<tr><td align=right>1/4 miil i sydost ett <td><b>bøsse-skud</b><td>
    breed i dend søe snavn (avsn 32599)</tr>
<tr><td align=right>1/8 miil laang og 2 <td><b>bøsseskud</b><td> breed
    og hænge i nord-væst (avsn 34618)</tr>
<tr><td align=right>3 bøssskud lang og 1 <td><b>bøsse-skud</b><td>
    breed snavn strækker sig fra (avsn 34618)</tr>
<tr><td align=right>hand meener er et par <td><b>bøsse-skud</b><td>
    breed uden skoug og græs (avsn 24089)</tr>
<tr><td align=right>nord i søer ett par <td><b>bøsse-skud</b><td> lang
    i snavn dette snavn (avsn 27903)</tr>
<tr><td align=right>hvoraf snavn 3 a 4 <td><b>bøsseskud</b><td> lang
    rinder i nord i (avsn 128044)</tr>
<tr><td align=right>en elv i syd-ost ett <td><b>bøsse-skud</b><td>
    lang ved nafn snavn j (avsn 32775)</tr>
<tr><td align=right>for dette snavn ett par <td><b>bøsse-skud</b><td>
    og 2 1/2 miil nord (avsn 34854)</tr>
<tr><td align=right>og spitz oven paa et <td><b>bøsse-skud</b><td>
    over af størrelse bart og (avsn 34854)</tr>

[...]
```

B.7.5 Skriv KWIC-konkordans sortert etter nodenummer

```
Valg: 5
```

```
Ordform: bøsseskud

<HTML><HEAD><TITLE>Konkordans bøsseskud</TITLE></HEAD>
<BODY>
<H1>Konkordans bøsseskud</H1>
<table>
<tr><td align=right>1/8 miil laang og 2 <td><b>bøsseskud</b><td> breed
    og hænge i nord-væst (avsn 34618)</tr>
<tr><td align=right>u-betydelig difference af et par
    <td><b>bøsseskud</b><td> og øie-siunet kan best skiønne (avsn
    51718)</tr>
<tr><td align=right>vii skarjaha-gorre et fieldskare 1
    <td><b>bøsseskud</b><td> østen for balvattenet fra sør (avsn
    53321)</tr>
<tr><td align=right>1/8 miil langt og 2 <td><b>bøsseskud</b><td> bredt
    hvoraf aaen rinder i (avsn 59065)</tr>
<tr><td align=right>i vester og en 5 <td><b>bøsseskud</b><td> bredt og
    det vestre vand (avsn 59112)</tr>
<tr><td align=right>øster 1/2 miil langt 2 <td><b>bøsseskud</b><td>
    over bredt hvoraf aaen rinder (avsn 59796)</tr>
<tr><td align=right>miil andre 2 a 3 <td><b>bøsseskud</b><td> bredt
    fladt og steen-uret med (avsn 59851)</tr>
<tr><td align=right>gaaer kiølen 5 à 6 <td><b>bøsseskud</b><td> i
    ost-nord-ost som sluttes fra (avsn 59983)</tr>
<tr><td align=right>fladagtigt ovenpaa 3 a 4 <td><b>bøsseskud</b><td>
    over stort fladt-nedstubendis sommestedz steen-uret (avsn 60049)</tr>
<tr><td align=right>jorden rinder en bæk 1 <td><b>bøsseskud</b><td>
    lang i syd-vest ad sverrig (avsn 60729)</tr>

[...]
```

B.8 Grunner

```
Valg: g

DELMENY: Grunner

1. Skriv ut alle grunner.
2. Skriv ut grunnene med tilknytning til en bestemt node.
3. Skriv ut alle grunner som er knyttet til avsnitt uttalt av talere av en kategori.
Q. Ut
```

B.8.1 Skriv ut alle grunner

```
Valg: 1

GRUNN 5368 av type Taler til avsnitt
     Årsak: Taler til et avnsitt (objekt 1) satt til et navnobjekt (objekt 2)
Signaturer
     Ansvarlig: parse-dokub: velg-taler-til-avsnitt
```

```
Tilknyttede objekter:
1. Kobling til node 129385 med tekst:
J Warangers Østbotten d. 15 February. 1745.
Peter Schnitler
Hendrich Mathisen
Lensmand (L. S.)
Peder Larsen i
Ostbotten (L. S.)
Tude Pedersen i
Ostbotten
2. Kobling til navn 3: Skal ikke analyseres
================================================================

GRUNN 5367 av type Taler til avnsitt
     Årsak: Taler til et avnsitt (objekt 1) satt til en node (objekt 2)
som er knyttet til et navnobjekt (objekt 3)

     Ansvarlig: parse-dokub: velg-taler-til-avsnitt

Tilknyttede objekter:
1. Kobling til node 129363 med tekst:
Efter tilspørgende sagde Vidnet Peder Minnesen at have hørt at i gl: tid for en 50 Aar
siden, skal Passvigs finner i Tallet have været vel 40 Familier som siden af Sott ere bortdøde
og formindskede til en halv snees Familier.
2. Kobling til node 129366 med tekst:
Peder Minnesen
3. Kobling til navn 354: Peder Minnesen
================================================================

[...]

GRUNN 2148 av type Kobling pnavn-node.
     Årsak: Soundex-kodingen av innholdet i noden (objekt 1)
er gjenfunnet i soundex-koding av en av navneformene i
pnavn-objektet (objekt 2). Sidene man leter på er hentet
fra sideinnførsler på objekt 2 etter vanlige regler.
     Ansvarlig: parse-dokub: legg-pnavn-paa-pnavnreg

Tilknyttede objekter:
1. Kobling til node 115476 med tekst:
Aamund Jonsen
2. Kobling til navn 4: Aamund Jonsen Lappe
================================================================

GRUNN 2147 av type Kobling pnavn-node.
     Årsak: Soundex-kodingen av innholdet i noden (objekt 1)
er gjenfunnet i soundex-koding av en av navneformene i
pnavn-objektet (objekt 2). Sidene man leter på er hentet
fra sideinnførsler på objekt 2 etter vanlige regler.
     Ansvarlig: parse-dokub: legg-pnavn-paa-pnavnreg

Tilknyttede objekter:
1. Kobling til node 72299 med tekst:
Aanet andersen
2. Kobling til navn 5: Aanud (Aanet, Aanod, Anud) Andersen
================================================================
```

```
[...]

GRUNN 1273 av type Navneformer på et navneobjekt
     Årsak: Navnet (objekt 1) finnes i *navn-navneformer*,
så navneformene legges inn derfra.
     Ansvarlig: parse-dokub: legg-inn-navneformer-paa-ett

Tilknyttede objekter:
1. Kobling til navn 4: Aamund Jonsen Lappe
================================================================

GRUNN 1272 av type Navneformer på et navneobjekt
     Årsak: Navnet (objekt 1) finnes i *navn-navneformer*,
så navneformene legges inn derfra.
     Ansvarlig: parse-dokub: legg-inn-navneformer-paa-ett

Tilknyttede objekter:
1. Kobling til navn 5: Aanud (Aanet, Aanod, Anud) Andersen
================================================================

[...]

GRUNN 828 av type Innførsel i navneregisteret
     Årsak: Første pnavn-node (objekt 1) i oppslagsnoden (objekt 2) lagt inn
som objekt i navneregisteret (objekt 3).
     Ansvarlig: parse-dokub: lag-navnereg-innforsel

Tilknyttede objekter:
1. Kobling til node 134260 med tekst:
&mdash; Olsen
2. Kobling til node 134717 med tekst:
&mdash; Olsen, fjellfinn i
Snåsa pgld 139, 147, 154, 156,
158 f. 161.
3. Kobling til navn 448: Zacharias Olsen
================================================================

GRUNN 827 av type Innførsel i tittelregister
     Årsak: Fant igjen første ord etter pnavn i tittel-kategoriregister.
     Ansvarlig: parse-dokub: finn-tittel

Tilknyttede objekter:
1. Kobling til navn 448: Zacharias Olsen
2. Kobling til node 134717 med tekst:
&mdash; Olsen, fjellfinn i
Snåsa pgld 139, 147, 154, 156,
158 f. 161.
================================================================

[...]
```

B.8.2 Skriv ut alle grunner med tilknytning til en bestemt node

```
Valg: 2

Nodeid: 133987
```

```
Grunner med tilknyting til node 133987:

GRUNN 785 av type Innførsel i navneregisteret
     Årsak: Første pnavn-node (objekt 1) i oppslagsnoden (objekt 2) lagt inn
som objekt i navneregisteret (objekt 3).
Fra bruker: Avvikende oppsett.
     Ansvarlig: parse-dokub: lag-navnereg-innforsel

Tilknyttede objekter:
1. Kobling til node 133987 med tekst:
&mdash;
Pedersen
2. Kobling til node 134694 med tekst:
&mdash;
Pedersen, sjøfinn i Varanger 330,
412.
3. Kobling til navn 425: Thomas (Thomes, Tomes) Pedersen
================================================================
```

B.8.3 Skriv ut alle grunner som er knyttet til avsnitt med talere av en kategori

```
Valg: 3

Kategori: reindriftssame

Grunner med tilknyting til node 110562:

GRUNN 4960 av type Taler til avnsitt
     Årsak: Taler til et avnsitt (objekt 1) satt til en node (objekt 2)
som er knyttet til et navnobjekt (objekt 3)

     Ansvarlig: parse-dokub: velg-taler-til-avsnitt

Tilknyttede objekter:
1. Kobling til node 110562 med tekst:
til Sp. 21. Svarer: at Arisbyes Finner siddet paa den Østre Side af Tanen-Elv, at for-
staae fra Karasjok-Kæften til Fossholmen, og ei kommet derover uden i deres FløtningsFærd,
med videre, som 48de Vidne pag. 369 forklaret.
2. Kobling til node 110506 med tekst:
Ammond Olsen
3. Kobling til navn 13: Ammond Olsen
================================================================

Grunner med tilknyting til node 110589:

GRUNN 4961 av type Taler til avsnitt
     Årsak: Taler til et avnsitt (objekt 1) satt til et navnobjekt (objekt 2)

     Ansvarlig: parse-dokub: velg-taler-til-avsnitt

Tilknyttede objekter:
1. Kobling til node 110589 med tekst:
Efter Tilspørgende om Raamerket i Tanen-Elv imellem de Norske Tanens- og fælles
Arisbyes Finner svarer han: De Arisbye-Finner sige, at Skaar-Aae, og Tanens Finner paa-
staae, at Fossholmen giør Skiellet.
2. Kobling til navn 13: Ammond Olsen
```

```
============================================================

Grunner med tilknyting til node 110624:

GRUNN 4962 av type Taler til avsnitt
     Årsak: Taler til et avnsitt (objekt 1) satt til et navnobjekt (objekt 2)

     Ansvarlig: parse-dokub: velg-taler-til-avsnitt

Tilknyttede objekter:
1. Kobling til node 110624 med tekst:
Sp. 22. Svar: Han veed om de Norske Refsbottens- og Porsangers Field-Finner, at de
sidde baade Norden og Sønden for Karasjok-Elven, som 48de Vidne p. 369 og de andre Vidner
pag. 374 ved dette Spørsmaal have udsagt. De Norske Laxefiords Field-Finner, da de vare
fleere i Mandtallet, have indtil for omtrent 6 Aar siden haft deres Leje til Tanen-Elv omtrent
imod Nulli-jok, Gieskadam og Otzjok-Munden, hvor han har seet dennem; Og at de samme
have gaaet over Tanen med deres Reen til Nullijok, og til under Gieskadam, naar Moesen
paa Nordre Side har skiortet, det har han hørt.
2. Kobling til navn 13: Ammond Olsen
============================================================

[...]
```

B.9 Lagring og tilbakehenting av datastruktur

Hele datastrukturen kan lagres, med unntakk av analysesttrukturene, men disse lages helautomatisk. Det viktigste er å kunne lagre alle datastrukturer som er laget ved hjelp av manuelle inngrep. Filnavn for SGML-filer og filer for lagring av databasen er hardkodet.

```
Valg: h

DELMENY: Lagring

1. Lagre datastrukturen.
2. Les inn lagret datastruktur.
Q. Ut
```